Gates On the Fly User Manual V1.2

- 1 Introduction
  - 1.1 Overview
  - 1.2 Netlist ECO Solutions
  - 1.3 Download and Install GOF
- 2 NetlistECO Flows
  - 2.1 Automatic FullLayers ECO Flow
    - 2.1.1 Overview
    - 2.1.2 Files and data requirements
    - 2.1.3 Steps to do automatic functional ECO
    - 2.1.4 Example GofCall scripts
    - 2.1.5 Run and debug
    - 2.1.6 Dedicated Logic Equivalence Check Engine
    - 2.1.7 Multiple modules in hierarchical netlist
    - 2.1.8 Uniquified modules
    - 2.1.9 Partial reference netlist
      - 2.1.9.1 Two steps ECO
      - 2.1.9.2 Step 1.Use partial netlist to fixpre-layoutnetlist
      - 2.1.9.3 Step2. Use the new reference netlist to fix implementation netlist
      - 2.1.9.4 Run two steps ECO
    - 2.1.10 Stitch new flops in toscan chain
    - 2.1.11 Flatten modules having flops
    - 2.1.12 Add a new module
    - 2.1.13 Note in RTL modification and re-synthesis
      - 2.1.13.1 Keep sequential signal name
      - 2.1.13.2 Use the same synthesis constraints
    - 2.1.14 Check design after ECO
    - 2.1.15 Exclude fix points in fix_modules
    - 2.1.16 Handle floating issue safter ECO
    - 2.1.17 Command'fix_modules' vs command'fix_logic'
  - 2.2 Automatic Metal Only ECO Flow
    - 2.2.1 Overview
    - 2.2.2 Files and data requirements
    - 2.2.3 Steps to do automatic Metal Only ECO
    - 2.2.4 Example GofCall script for Metal Only ECO
    - 2.2.5 Run and debug
    - 2.2.6 Gated clocks in AutomaticMetal Only ECO
  - 2.3 Timing ECO Flow
    - 2.3.1 Overview
    - 2.3.2 Files and data requirements
    - 2.3.3 Procedures
      - 2.3.3.1 Read in database
      - 2.3.3.2 Parse Primetime Report File
      - 2.3.3.3 Check Path on LayoutViewer
      - 2.3.3.4 ECO Operations
  - 2.4 Script Mode Full Layers Manual ECO Flow
    - 2.4.1 Overview
    - 2.4.2 Steps to do Manual ECO In Scripts
    - 2.4.3 Locate ECO point
    - 2.4.4 Files and data requirements
    - 2.4.5 ECO APIs list
    - 2.4.6 Example GofCall script for Manual ECO
    - 2.4.7 Run and debug
    - 2.4.8 Handle repetitive work
    - 2.4.9 Special character
  - 2.5 Script Mode Metal Only Manual ECO Flow
    - 2.5.1 Overview
    - 2.5.2 Files and data requirements
    - 2.5.3 Example GofCall script for Manual Metal Only ECO
    - 2.5.4 Run and debug
  - 2.6 GUI Mode Full Layers ECO Flow
    - 2.6.1 Overview
    - 2.6.2 Start up GOF in GUI Mode
    - 2.6.3 Create Partial Schematic
    - 2.6.4 Do ECO on schematic
    - 2.6.5 Save ECO
  - 2.7 GUI Mode Metal Only ECO Flow
    - 2.7.1 Overview
    - 2.7.2 Methods for Metal Only ECO
    - 2.7.3 Setup and use cases
- 3 Guidance to choose Netlist ECO flows
  - 3.1 Overview
  - 3.2 Decision Factors
  - 3.3 ECO flows decision tree
- 4 Script Mode Detail Features
  - 4.1 GofCall Script
    - 4.1.1 Get Help
    - 4.1.2 Feature list
    - 4.1.3 API list
    - 4.1.4 API sgrouping
      - 4.1.4.1 Netlist Browse APIs
      - 4.1.4.2 Automatic ECO APIs
      - 4.1.4.3 File IO APIs
      - 4.1.4.4 Manual ECO APIs
    - 4.1.5 API usage
  - 4.2 String Handling In Script Mode
    - 4.2.1 Single quote and double quote
    - 4.2.2 Instance and netwith backslash
  - 4.3 Run and debug GofCall script
    - 4.3.1 Command line
    - 4.3.2 GOF Shell
    - 4.3.3 Run in GUI mode
    - 4.3.4 Fast schematic launch
    - 4.3.5 Break points for debug
  - 4.4 Typical Manual ECO operations
    - 4.4.1 Insert gate to port
      - 4.4.1.1 Insert an invert to input port
      - 4.4.1.2 Insert to output port
      - 4.4.1.3 Insert inverts to multiple ports
    - 4.4.2 Insert gate to register instance pin
      - 4.4.2.1 Insert invert to flop data pin
      - 4.4.2.2 Insert invert to flop output pin
      - 4.4.2.3 Insert MUX to data pin of multiple flops
    - 4.4.3 Change flops to other type
      - 4.4.3.1 Changenon-resetflop type to resettable flop
    - 4.4.4 Insert gate to hierarchical instance pin
      - 4.4.4.1 Insert invertsto hierarchical instance pins
      - 4.4.4.2 Insert AND to hierarchical instance pins
- 5 GUI ModeDetail Features
  - 5.1 GofViewer
    - 5.1.1 Log Window
    - 5.1.2 File Menu
      - 5.1.2.1 Load Design
      - 5.1.2.2 Reload Design
      - 5.1.2.3 Save Startup File
      - 5.1.2.4 Open Other Netlist
      - 5.1.2.5 Open Log Window
      - 5.1.2.6 Exit
    - 5.1.3 Find Menu
      - 5.1.3.1 Search
      - 5.1.3.2 Goto Line Number
      - 5.1.3.3 ReportArea
      - 5.1.3.4 Report Leakage
      - 5.1.3.5 Report Leaf Cells
      - 5.1.3.6 Report Submodules
      - 5.1.3.7 Statistic of Current Design
      - 5.1.3.8 List Library
      - 5.1.3.9 List Context for Leaf Cell
    - 5.1.4 Commands Menu
      - 5.1.4.1 Launch GofTrace Schematic
      - 5.1.4.2 Launch GofTrace with Gate
      - 5.1.4.3 Launch Layout Viewer
      - 5.1.4.4 Launch GofCall Script Interface
      - 5.1.4.5 Switch to GOF Shell Mode
      - 5.1.4.6 Spare Cells
      - 5.1.4.7 Launch Files Compare Window
      - 5.1.4.8 Process Timing Violations
      - 5.1.4.9 SDF

- 5.1.5 Options Menu
  - 5.1.5.1 Hierarchy Window Font
  - 5.1.5.2 Netlist Window Font
  - 5.1.5.3 Dump Waveform Restore File
  - 5.1.5.4 Setup
- 5.1.6 Help Menu
  - 5.1.6.1 General
  - 5.1.6.2 About
  - 5.1.6.3 nandigits.com/gof_manual.htm
  - 5.1.6.4 Read Ethernet Mac Address
- 5.1.7 Keyboard Shortcuts
  - 5.1.7.1 Access Menu
  - 5.1.7.2 Functions access
- 5.1.8 Selection Status
- 5.1.9 Netlist Window Pop Menu
  - 5.1.9.1 Search
  - 5.1.9.2 Copy Selected to
  - 5.1.9.3 Driver of the selected net
  - 5.1.9.4 List Connectivity of the selected net
  - 5.1.9.5 List Fanin EndPoints
  - 5.1.9.6 List Fanout EndPoints
  - 5.1.9.7 Parent Module
  - 5.1.9.8 List Context
- 5.1.10 Hierarchy Window Pop Menu
  - 5.1.10.1 Show Definition
  - 5.1.10.2 Show Calling
  - 5.1.10.3 Report Area of the selected design
  - 5.1.10.4 Report Leakage of the selected design
  - 5.1.10.5 Report Leaf Cells of the selected design
  - 5.1.10.6 Report Submodules of the selected design
  - 5.1.10.7 Statistic of the selected design
  - 5.1.10.8 Edit Module of the selected design
  - 5.1.10.9 Save Module of the selected design
  - 5.1.10.10 Goto Line Number#

5.2 GofTrace

- 5.2.1 Mouse buttons usage
  - 5.2.1.1 Mouse Left Button
  - 5.2.1.2 Mouse Middle Button
  - 5.2.1.3 Mouse Right Button
- 5.2.2 File Menu
  - 5.2.2.1 Save
  - 5.2.2.2 Open
  - 5.2.2.3 Print
  - 5.2.2.4 Exit
- 5.2.3 Schematic Menu
  - 5.2.3.1 New Schematic
  - 5.2.3.2 List Gate
  - 5.2.3.3 Load Gate
  - 5.2.3.4 Load Gate Driving Net
  - 5.2.3.5 List Selected Instances
  - 5.2.3.6 List Selected Wires
  - 5.2.3.7 List Selected Modules
  - 5.2.3.8 List Selected Instances Definitions
  - 5.2.3.9 List Selected Gates Types
  - 5.2.3.10 Zoom In
  - 5.2.3.11 Zoom Out
  - 5.2.3.12 Zoom to
  - 5.2.3.13 Find Gates on Schematic
  - 5.2.3.14 Find Nets on Schematic
  - 5.2.3.15 Undo Schematic Operations
  - 5.2.3.16 Place & Route
  - 5.2.3.17 Create PS/PDF File
- 5.2.4 Commands Menu
  - 5.2.4.1 View Gates in Layout
  - 5.2.4.2 Load Layout Files
  - 5.2.4.3 Launch GofCall Script Interface
  - 5.2.4.4 Spare Cells
  - 5.2.4.5 Launch Files Compare Window
  - 5.2.4.6 Process Timing Violations
  - 5.2.4.7 SDF
- 5.2.5 Options Menu
  - 5.2.5.1 Increase Font Size
  - 5.2.5.2 Decrease Font Size
  - 5.2.5.3 Show Port
  - 5.2.5.4 Show Wire
  - 5.2.5.5 Show Title
  - 5.2.5.6 Show Type
  - 5.2.5.7 Show Connections
  - 5.2.5.8 Show Comment
  - 5.2.5.9 Dump Waveform Restore File
  - 5.2.5.10 Save String to Clipboard
  - 5.2.5.11 Cursor Mode
  - 5.2.5.12 Line Edit Mode
  - 5.2.5.13 Setup
- 5.2.6 Help Menu
  - 5.2.6.1 General
  - 5.2.6.2 About
  - 5.2.6.3 nandigits.com/gof_manual.htm
- 5.2.7 Keyboard Shortcuts
  - 5.2.7.1 Access Menu
  - 5.2.7.2 Functions access
- 5.2.8 Selection Status
- 5.2.9 GofTrace Pop Menu
  - 5.2.9.1 Nets Equivalence Check
  - 5.2.9.2 Find Logic Equivalent Nets
  - 5.2.9.3 Find Circuit between Two Points
  - 5.2.9.4 Add Comments
  - 5.2.9.5 Find Gates on Schematic
  - 5.2.9.6 Find Nets on Schematic
  - 5.2.9.7 Place & Route
  - 5.2.9.8 Show Logic Cone
  - 5.2.9.9 Copy Selected to
  - 5.2.9.10 Find selected in GofViewer
  - 5.2.9.11 Edit Gate Display
  - 5.2.9.12 List Logic for the Selected Leaf Cell
  - 5.2.9.13 List Context for the Selected Leaf Cell
  - 5.2.9.14 List Definition for the Selected Instance
  - 5.2.9.15 Load Instance Similar to the Selected Instance
  - 5.2.9.16 Equivalent Symbol
  - 5.2.9.17 Delete

5.3 GofECO

- 5.3.1 ECO Menu
  - 5.3.1.1 Enable ECO and ECO Preferences
  - 5.3.1.2 Insert Gates
  - 5.3.1.3 Replace Gates
  - 5.3.1.4 Add Gates
  - 5.3.1.5 Delete Selected Items
  - 5.3.1.6 Upsize Drive Strength
  - 5.3.1.7 Downsize Drive Strength
  - 5.3.1.8 Undo ECO Operations
  - 5.3.1.9 Add Connection
  - 5.3.1.10 Save ECO
- 5.3.2 MetalOnlyECO
  - 5.3.2.1 Metal ECO, mode 1
  - 5.3.2.2 Metal ECO, mode 2
  - 5.3.2.3 Metal ECO, mode 3
  - 5.3.2.4 Metal ECO, mode 4

5.4 LayoutViewer

- 5.4.1 File Menu
  - 5.4.1.1 Capture in PDF
  - 5.4.1.2 Exit
- 5.4.2 Commands Menu
  - 5.4.2.1 Clear Circuit Markers
  - 5.4.2.2 Clear Search Markers
  - 5.4.2.3 New Schematic
- 5.4.3 OptionsMenu
  - 5.4.3.1 Show Grid
  - 5.4.3.2 Show Instance
  - 5.4.3.3 Show Module
  - 5.4.3.4 Setup

5.4.4 Help Menu
5.4.4.1 Help on LayoutViewer
5.4.5 LayoutViewer Pop Menu
5.4.5.1 Clear Circuit Markers
5.4.5.2 Clear Searching Markers
5.4.5.3 Copy Selected to
5.4.6 Keyboard and mouse combination
5.4.6.1 Ctrl key to measure length
5.4.6.2 Shift key to select multiple markers
5.4.7 Mouse operations
5.4.8 Select color buttons
5.4.9 Search function
6 Appendix A
6.1 GOF Command Options
6.2 Command line Examples
7 Appendix B
7.1 Fatal codes
7.2 Error codes:
7.3 Warning codes
7.4 GUI warning codes

1 Introduction

1.1 Overview

Gates On the Fly, GOF, provides complete netlist solutions to accommodate various netlist ECO and debug scenarios.

- Complete Netlist ECO solutions.
- Fully Interactive Schematic.
- Netlist Lint Checker.
- Flexible Netlist Database Report.
- Script Interface for Tool Extensions.

1.2 Netlist ECO Solutions

Gates On the Fly implements several cutting edge ECO methodologies. Netlist ECO varies in size and complexity from case to case. Design process changes from company to company. Gates On the Fly gives users flexibility to choose one or several of the methodologies depending on the size and complexity of the changes.

- Automatic ECO and Manual ECO.
- Script Mode and GUI Mode.
- Metal Only ECO and All Layers ECO.
- Functional ECO and Timing ECO.

- **Automatic mode ECO**

The automatic functional ECO is done by a GofCall script. The flow needs Implementation Netlist which is under ECO, and Reference Netlist which is re-synthesized from modified RTL with the same constraints as the pre-layout netlist. Users specify the modified module names in 'fix_modules' API. GOF uses the built-in Logic Equivalence Check engine to analyze the listed modules to find the non-equivalent points. Logic patches are created to fix the modules. The final patches are optimized circuits with minimum gate count that make Implementation Netlist equal to Reference Netlist. The patches can be mapped to spare-type-gates by 'map_spare_cells' API.

Timing ECO can be done in automatic mode as well. Primetime report file is needed in automatic timing ECO.

- **Manual mode ECO**

When ECO changes are known and ECO size is small or the operations are repetitive like adding inverts on a bus, manual mode ECO is better choice. Since it is more efficient and the final gates touched can be less than automatic mode ECO. Moreover, automatic and manual mode ECO can be interleaved in one GofCall ECO script.

- **Metal Only ECO**

When ECO is done in either automatic mode or manual mode, 'map_spare_cells' command is run to convert the newly added cells to spare gate types cells. Users can control only spare gate type cells being used in manual mode ECO, so that the converting stage can be bypassed.

- **Hierarchical ECO**

GOF supports hierarchical ECO by set the ECO scope the sub-modules. Some Logic Equivalence Check cases can only be resolved in flatten mode. Since GOF only focuses on the modules or spots that user specifies, it can avoid to fall into the trap of LEC failing in hierarchical netlist.

- **Timing ECO**

GOF reads in Primetime report file and does bottleneck analysis of the violations. The critical path is automatically fixed. User can choose to do timing ECO manually with ECO APIs.

- **GUI mode ECO**

GUI mode ECO has advantage of fast ramping up. It's good for small size ECO. The incremental schematic feature is very helpful for analyzing the netlist before the next step is decided.

- **Integrated environment**

The incremental schematic is very useful in pinpointing the logic issues. GOF has exports many useful APIs to fast access the database.

In section 2, eight ECO flows are listed to accommodate different ECO scenarios. In section 3, a guideline is provided for choosing the ECO flow.

1.3 Download and Install GOF

Gates On the Fly releases can be found in

http://www.nandigits.com/download.htm

Three platforms are supported, MS-Windows, Linux 32bits OS and Linux 64bits OS. Download the proper platform release file, and unzip to a directory. Set 'the_install_path/GOF/bin' in search path.

Contactsupport@nandigits.comfor evaluation license, if the evaluation netlist size is larger than 4M bytes.

2 NetlistECO Flows

2.1 Automatic FullLayers ECO Flow

2.1.1 Overview

Full layers functional ECO can add or delete gates freely. The ECO operations are done in a GofCall script which is compatible with Perl, and it uses exported APIs to access the netlist database. GOF reads in two netlist files, Implementation Netlist which is under ECO and Reference Netlist which is re-synthesized from modified RTL with the same constraints as the pre-layout netlist. In the GofCall script, the modified modules are specified in 'fix_modules' API. GOF uses the built-in Logic Equivalent Check Engine to figure out the non-equivalent points in the specified modules. And optimized minimum size gate patches are applied to fix the non-equivalent modules.

Figure1 shows two logic cones from the implementation and reference netlist for the same compare point. The implementation point mismatches the reference point initially. GOF compares the two points and extracts the patch logic from Reference Netlist and applies to Implementation Netlist. After the patching, the two points become equivalent.

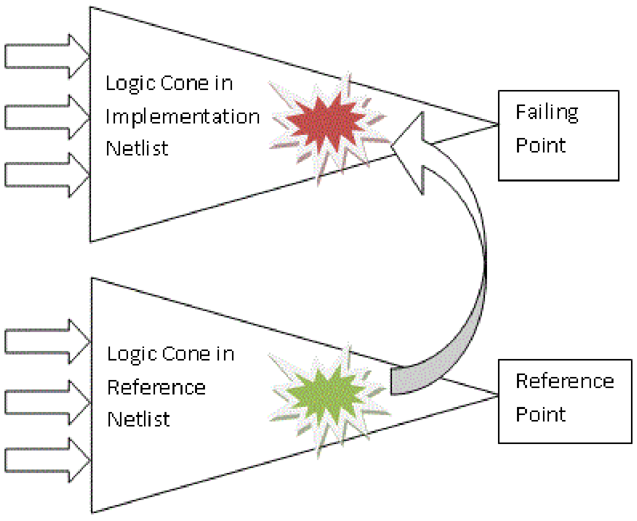


Figure 1 Logic Cone Optimization

GOF does logic cone analysis and optimization for each failing point in the modules user specifies. The failing point is in format of output port or sequential element's input pin, such as flop's D input. User can specify the failing points directly instead of the module name to avoid redundant fixes and speed up the process. The final patch has the minimum number of gates to make the implementation logic cone equal to the reference logic cone.

The flow chart is shown in Figure 2.

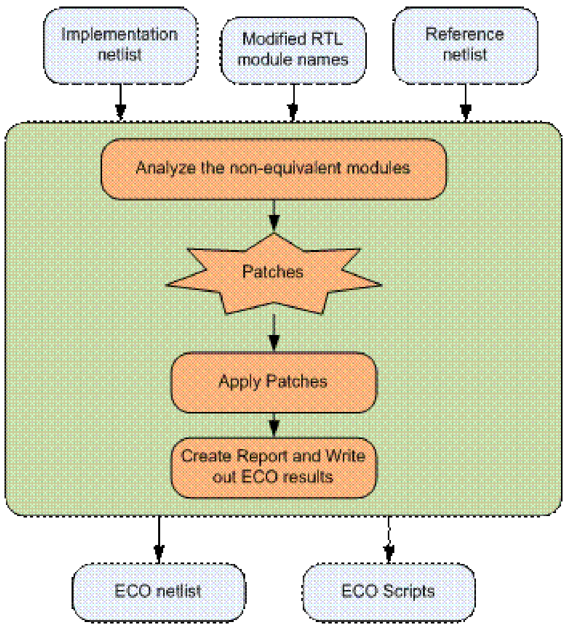

Implementation netlist
Modified RTL module names
Reference netlist
Analyze the non-equivalent modules
Patches
Apply Patches
Create Report and Write out ECO results
ECO netlist
ECO Scripts

Figure 2, Automatic functional ECO flow

2.1.2 Files and data requirements

- Standard library (Synopsys Liberty) files with extension '.lib'.
- Other Verilog libraries files if not covered in '.lib' files.
- Implementation Netlist on which ECO will be done.
- Reference Netlist synthesized with the same constraints as the pre-layout netlist.
- The modified module names.

2.1.3 Steps to do automatic functional ECO

A typical situation for an automatic functional ECO:

- Modify RTL modules "module_A", "module_B" and "module_C"
- Synthesize the new RTL to get Reference Netlist
- Create a GofCall ECO script:
  - Specify ECO name in 'setup_eco'
  - Load Standard Cell libraries and Verilog libraries
  - Load Reference Netlist and Implementation Netlist
  - Add fix command 'fix_modules("module_A", "module_B", "module_C");'
  - Report ECO status and write out ECO results
- Run the above script

2.1.4 Example GofCall scripts

The GofCall script has exact the same syntax of Perl script. But it can execute exported APIs that access the netlist database and modify the netlist.

The following is an example script for an automatic functional ECO:

```
# GofCall ECO script, run_example.pl
use strict;
undo_eco;# Discard previous ECO operations
setup_eco("eco_example");# Setup ECO name
read_library("art.90nm.lib");# Read in standard library
read_design("-ref", "reference.gv");# Read in Reference Netlist
read_design("-imp", "implementation.gv");# Read in Implementation Netlist Which is under ECO
fix_modules("module_A", "module_B", "module_C");# Specify the modules that have been modified
set_top("topmod");# Set the top module
report_eco(); # ECO report
check_design("-eco");# Check if the ECO causes any issue, like floating
write_verilog("eco_verilog.v");# Write out ECO result in Verilog
write_soce("eco_soce.tcl");# Write out TCL script for SOC Encounter
exit;# Exit when the ECO is done, comment it out to go to interactive mode when 'GOF >' appears
```

2.1.5 Run and debug

The GofCall Script can be run by '-run' option.

gof -run run_example.pl

CheckRun and debug GofCall script sectionfor more detail

2.1.6 Dedicated Logic Equivalence Check Engine

The built-in Logic Equivalence Check Engine is dedicated to search equivalent nets in Implementation Netlist to optimize the patch circuit. The searching process is global.

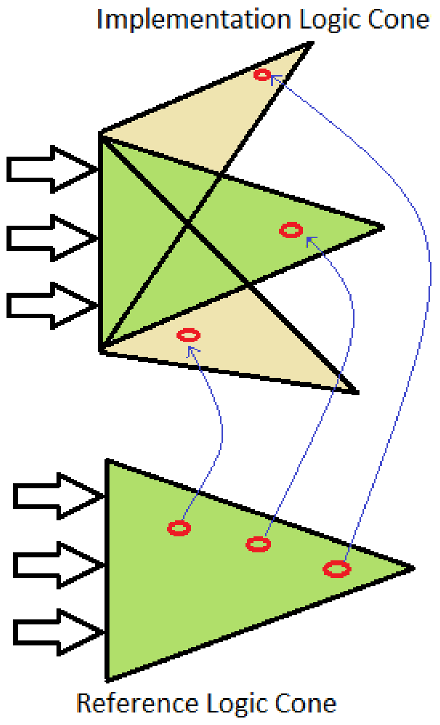


Figure 3 Dedicated LEC Engine

For each net in the Reference Logic Cone, any net in the Implementation Logic Cone with the same fanin end-points will be compared for Logic Equivalence.

Users can check equivalence on any two nets in the reference and implementation netlists. The API 'comare_nets' can be used to compare any two nets in Reference Netlist and Implementation Netlist.

Compare_nets API:

Check equivalence of two nets in the reference and implementation netlist

```
Usage: my $result = compare_net($net0, $net1);
$net0: The net in Reference Netlist.
$net1: The net in Implementation Netlist.
$result: If 1, they are equal, if 0, they are not equal.
Examples:
# Compare reg1/D in the reference and reg1/D in Implementation Netlist
compare_nets("reg1/D", "reg1/D");
```

Since the Logic Equivalence Check engine compares every net in the logic cone, it makes very easy to debug logic equivalence mismatches in two netlists. Read two nets equivalence check in GUI mode for more detail.

2.1.7 Multiple modules in hierarchical netlist

In hierarchical netlist, every module having failing point should be put into 'fix_modules' arguments. Any boundary changes should have both the sub-module and parent-module put into 'fix_modules' command. GOF fixes the modules in the sequence as the modules being entered.

For example, module 'top' has sub-module 'smod'. 'smod' has flop 'state_reg_0_' changed and new input port 'in0' added. The fix command should be

fix_modules("smod", "top");

GOF will fix module 'smod' first by adding input port 'in0' and fixing fanin of the flop 'state_reg_0_'. Then module 'top' is fixed to add driving network to drive input port 'in0' of sub-module 'smod'.

Adjusting the fix sequence would affect the final result slightly, though the ECOed netlists are equivalent to Reference Netlist in different fix sequences.

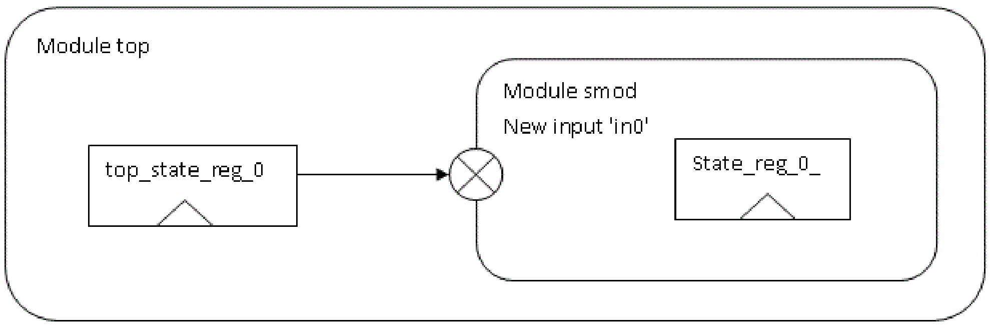


Figure 4 Across multiple modules

```
# GofCall ECO script, run_hier_example.pl
use strict;
undo_eco;# Discard previous ECO operations
setup_eco("eco_hier_example");# Setup ECO name
read_library("art.90nm.lib");# Read in standard library
read_design("-ref", "reference.gv");# Read in Reference Netlist
# Read in Implementation Netlist which is under ECO
read_design("-imp", "implementation.gv");
fix_modules("smod", "top");
set_top('top');# Set the scope top module
report_eco(); # ECO report
check_design("-eco");# Check if the ECO causes any issue, like floating
write_verilog("eco_verilog.v");# Write out ECO result in Verilog
write_soce("eco_soce.tcl");# Write out TCL script for SOC Encounter
```

2.1.8 Uniquified modules

If a module has multiple instances, after uniquify synthesis command, each instance has a new unique module name which is normally a string of '_' and numbers as suffix. 'get_modules' API can be used to get all uniquified modules and add them into 'fix_modules' argument list. Note, wildcard '*' is used in 'get_modules' API and '-hier' option is to find all matching modules in the design globally.

```
# GofCall ECO script, run_hier_example.pl
use strict;
undo_eco;# Discard previous ECO operations
setup_eco("eco_hier_example");# Setup ECO name
read_library("art.90nm.lib");# Read in standard library
read_design("-ref", "reference.gv");# Read in Reference Netlist
# Read in Implementation Netlist which is under ECO
read_design("-imp", "implementation.gv");
set_top('top');
my @tx_mods = get_modules("tx_controller*", "-hier");
my @rx_mods = get_modules("rx_controller*", "-hier");
fix_modules(@tx_mods, @rx_mods);
set_top('top');# Set the scope top module
report_eco();# ECO report
check_design("-eco");# Check if the ECO causes any issue, like floating
write_verilog("eco_verilog.v");# Write out ECO result in Verilog
```

2.1.9 Partial reference netlist

2.1.9.1 Two steps ECO

In some design flows, a full chip synthesis may take too long time. To improve the turn-around time, only these modules involving ECO are re-synthesized. However the partial netlist can't directly apply to post-layout Implementation Netlist. Since there maybe boundary optimization in the post-layout Implementation Netlist, partial netlist doesn't have full boundary information, so two steps ECO is used.

- Step 1. The partial new reference netlist is to fix the pre-layout netlist. And new full reference netlist is generated by the partial netlist. 'set_mod2mod' is used to do module mapping in this step.
- Step2. The new full reference is used to fix Implementation Netlist.

2.1.9.2 Step 1.Use partial netlist to fixpre-layoutnetlist

Code example:

```
# eco_step1.pl
read_library("art.90nm.lib");# Read in standard library
read_design("-ref", "partial_reference.gv");# Read in the partial Reference Netlist
# Read in the pre-layout reference netlist under ECO
read_design("-imp", "pre_layout.gv");
# Map 'smod' in partial netlist to 'smod' in the full netlist under ECO
set_mod2mod("smod", "smod");
set_top("smod");
fix_modules("smoda", "smodb", "smod");
set_top("top");
check_design("-eco");# Check if the ECO causes any issue, like floating
write_verilog("new_reference.gv");
exit;
```

2.1.9.3 Step2. Use the new reference netlist to fix implementation netlist

Code example:

```
# eco_step2.pl
read_library("art.90nm.lib");# Read in standard library
read_design("-ref", "new_reference.gv");# Read in the partial Reference Netlist
# Read in Implementation Netlist under ECO
read_design("-imp", "implementaion.gv");
fix_modules("smoda", "smodb", "smod");
set_top("top");
check_design("-eco");# Check if the ECO causes any issue, like floating
write_verilog("eco_implementation.gv");
exit;
```

2.1.9.4 Run two steps ECO

GOF will be run two times to get final ECOed netlist "eco_implementation.gv"

```
gof -run eco_step1.pl
gof -run eco_step2.pl
```

2.1.10 Stitch new flops in toscan chain

Scan chain can be updated to insert the new flops. 'stitch_scan_chain' command can be used to automatically stitch up the new flops.

For example, eight new flops 'state_new_reg_0' to 'state_new_reg_7' are added in fix_modules command. To insert them to the scan chain before 'pulse_reg'

```
stitch_scan_chain('-to', 'pulse_reg');
```

The scan chain can be reconnect up by manual change_pin commands as well.

2.1.11 Flatten modules having flops

If a module/instance hierarchy is in Reference Netlist, not in Implementation Netlist. When the module has pure combinational logic without sequential gates (flops) inside, GOF will flatten the module automatically. If the module has sequential gates, GOF will print error message

ERROR! (E-067B) Sequential instance 'instance_name' doesn't exist in module 'module_name' in IMP

In this case, 'flatten_modules' should be run to flatten the module. However, it is highly recommended that RTL and synthesis constraints should be handled properly that the reference/implementation netlists have the exact same hierarchies, since flattening a large module would deteriorate the ECO quality, lots of redundant new gates being added.

```
# GofCall ECO script, run_flatten_module_example.pl
use strict;
undo_eco;# Discard previous ECO operations
setup_eco("eco_hier_example");# Setup ECO name
read_library("art.90nm.lib");# Read in standard library
read_design("-ref", "reference.gv");# Read in Reference Netlist
# Read in Implementation Netlist which is under ECO
read_design("-imp", "implementation.gv");
flatten_modules("syn_macro", "retime", "retime_2");
fix_modules("smod", "top");
set_top('top');# Set the scope top module
report_eco();# ECO report
check_design("-eco");# Check if the ECO causes any issue, like floating
write_verilog("eco_verilog.v");# Write out ECO result in Verilog
write_soce("eco_soce.tcl");# Write out TCL script for SOC Encounter
```

2.1.12 Add a new module

The module mentioned in the section above can have hierarchy kept instead of flatten, and being written into ECO netlist as whole. This flow needs the module and its sub-modules written out in a separate verilog file, then uses read_library to load the file with '-vmacro' option. GOF will treat the module as a leaf cell.

An example for adding a new module:

```
# GofCall ECO script, run_new_module_example.pl
use strict;
undo_eco;# Discard previous ECO operations
setup_eco("eco_hier_example");# Setup ECO name
read_library("art.90nm.lib");# Read in standard library
read_library("-vmicro", "syn_macro.v");
read_design("-ref", "reference.gv");# Read in the Reference Netlist
# Read in the implementation netlist which is under ECO
read_design("-imp", "implementation.gv");
fix_modules("smod", "top");
set_top('top');# Set the scope top module
report_eco();# ECO report
check_design("-eco");# Check if the ECO causes any issue, like floating
write_verilog("eco_verilog.v");# Write out ECO result in Verilog
write_soce("eco_soce.tcl");# Write out TCL script for SOC Encounter
```

The content in file syn_macro.v will be written into ECO file eco_verilo.v as a whole. The corresponding instance will be created as well with ports connected correctly according to Reference Netlist.

2.1.13 Note in RTL modification and re-synthesis

When modifying RTL and do re-synthesis, care should be taken to maintain the database as much alike Implementation Netlist as possible.

2.1.13.1 Keep sequential signal name

A common problem in modifying RTL is having sequential signal name changed, which will appear in Reference Netlist as a different flop instance. For example

```
always @(posedge clk) abc <= abc_next;
```

It will create a flop instance 'abc_reg' in synthesis. If the ECO in RTL change this to

```
always @(posedge clk) abc_new <= abc_next;
```

After synthesis, a new flop instance 'abc_new_reg' would be created, which will bring a redundant fix in the new register creation.

So it is highly recommended to keep the sequential signal names.

2.1.13.2 Use the same synthesis constraints

When do re-synthesis, the same constraints should be used as what has been used in Implementation Netlist synthesis. If any hierarchy is not present in Implementation Netlist, flattening command should be used in synthesis to flatten the module to the maintain the same hierarchies.

2.1.14 Check design after ECO

It is highly recommended to run 'check_design' after ECO, to speed up, users can specify '-eco' option,

```
check_design('-eco')
```

It can detect if there is any floating or multiply drivers after ECO.

2.1.15 Exclude fix points in fix_modules

Some fixes are not necessary in 'fix_modules' API, for example, test related input pins and output ports. They can be excluded from being fixed by '-exclude' option.

For example, the following module has test related output port, 'scan_out_123' and 'scan_out_456', and it has test signals 'scan_en_0', 'scan_en_1' and 'scan_in_123' driving sub hierarchical instance 'u_subabc'.

```
module abc(scan_en, scan_in, scan_out_123,scan_out_456, in0, out0…);
input scan_en, scan_in, scan_out _123, scan_out_456;
input [7:0] in0;
output [7:0] out0;
wire scan_en_0_buf, scan_en_1_buf, scan_in_123_buf;
…
Subabc u_subabc(.scan_en_0(scan_en_0_buf), .scan_en_1(scan_en_1_buf),
.scan_in_123(scan_in_123_buf),
…);
…
endmodule
```

Users can exclude them from being fixed by '-exclude'

```
fix_modules('abc', '-exclude', 'scan_out_*,scan_in_*,scan_en_*');
```

Note, there is no need to specify path for 'scan_in_*' and 'scan_en_*'.

2.1.16 Handle floating issue safter ECO

After ECO, 'check_design' should be run and it can catch floating issues caused by the ECO. The reasons for floating can be:

1. Missed a module in 'fix_modules' argument
2. Unresolved instances caused by missing library support
3. Backend tool creating clone input port, making the original unused and floating

For item 1, the solution is simple, putting the missing module into 'fix_modules' argument should resolve the issue.

For Item 2, the solution is simple, putting the missing library in 'read_library' API should resolve the issue.

For Item 3, putting the module with floating nets into 'fix_modules' argument should resolve the issue. There is another way for the problem, using '-exclude' option in 'fix_modules' API. The assumption is the logic cone driven by the floating input port has no need to fix, so that the end fanout logic points of the floating input port can be put into '-exclude' option.

```
fix_modules('abc_mod', '-exclude', "state_reg_0/D,state_reg_1/D,scan_out*");
```

2.1.17 Command'fix_modules' vs command'fix_logic'

The command 'fix_modules' takes module names as arguments. GOF figures out the non-equivalent logic in the module and fix the logic. While, user needs to specify the exact fix points in 'fix_logic' command which can be output ports, flop instances , flop input pins , hierarchical instance input pins or block box input pins.

Apparently, 'fix_logic' is faster than 'fix_modules' command. If user is aware of which logic being modified, the modified points can be directly put into 'fix_logic' argument to speed up turn around time.

For example,

```
fix_modules("abc_mod")
```

The command is to fix some modified logic in module 'abc_mod'. If user knows that the modified logic is actually output port 'lms_working' and flops 'sm_state_reg_0', 'sm_state_reg_1'. The command can be replaced by

```
fix_logic("lms_working", "sm_state_reg_0", "sm_state_reg_1")
```

Note, 'fix_logic' should be paired with 'set_top' command to set the fix scope.

```
set_top("abc_mod");
fix_logic("lms_working", "sm_state_reg_0", "sm_state_reg_1");
```

2.2 Automatic Metal Only ECO Flow

2.2.1 Overview

In Metal Only ECO, the design has completed place and route. Any new gates added should map to spare gates that located in the design.

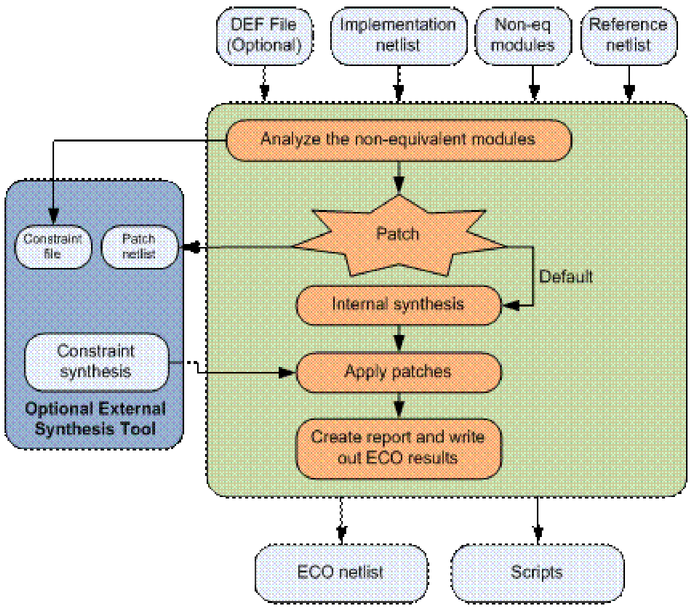


Figure 5, Metal Only ECO

The flow can use internal synthesis engine or external synthesis tool to map patch logic to spare types. It requires one of the spare type combinations as the following

- Two ports 'and/or' gates and 'inv' gates, 'mux' is optional.
- Two ports 'nand/nor' gates and 'inv' gates, 'mux' is optional.

A Design Exchange Format file is needed to map new instances to the closest spare gate instances. If DEF file is not loaded, GOF processes the ECO with gates type from the spare list without mapping to the exact spare instances. P&R tool like SOC Encounter will map the new instances in new netlist to the closest spare gates.

In 'fix_modules' command, GOF analyzes the modules to isolate the non-equivalent points and optimize the logic cone to find the minimum gate count patch circuit.

The flow can use external Synthesis Tool as well. The executable synthesis command should be in the search path. The supported Synthesis Tool is RTL Compiler from Cadence and Design Compiler from Synopsys.

GOF writes out the patch in Verilog file and a TCL script for external Synthesis Tool if it's enabled. The TCL script is to constrain the Synthesis Tool to use spare gates only when remapping the gates in the patch file. The Synthesis Tool is run with the Verilog file and TCL script as inputs, and it writes out remapped Verilog patch file which has only spare gate types.

When the spare-only patch file is created, user can pause the flow by '-pause' in 'map_spare_cells' command. User can either modify the patch file manually or tunes up the constraint file to rerun the synthesis for several iterations until the patch netlist meets the requirement. Then press 'n' key to resume the flow.

GOF reads back the spare-only patch file and fits the circuit into Implementation Netlist to fix the logic cones.

When ECO is done, a report can be created and ECO netlist/ECO script can be written out for the back end tool and LEC tool.

2.2.2 Files and data requirements

- Standard library (Synopsys Liberty) files with extension '.lib'.
- Other Verilog libraries if '.lib' files can't cover.
- Implementation Netlist.
- Reference Netlist.
- DEF (Design Exchange Format) file. It's optional. If it is not loaded, GOF won't map the spare gate type cells to the exact spare instances.
- Spare gates pattern. It is in 'hierarchical_instance/leaf_instance' format. It has wild card '*' to match the spare gates in Implementation Netlist.
- Spare gates list file. If several users work on the same Implementation Netlist, the initial spare gates list file should be generated only once. And new spare gates list file should be created every time an ECO is done.
- Modules that have been modified.

2.2.3 Steps to do automatic Metal Only ECO

A typical process for an automatic Metal Only ECO:

- Change RTL modules
- Synthesize the new RTL to get Reference Netlist
- Draft a GofCall script:
- Load Standard Cell libraries and Verilog libraries.
- Load Reference Netlist and Implementation Netlist.
- Specify the non-equivalent modules in 'fix_modules' command.
- Load DEF file, optional.
- Load LEF file, optional. It's useful in LayoutViewer feature.
- Create Spare Gates List by Spare Gates pattern or by reading in spare list file.
- Run 'map_spare_cells' to remap the patch from 'fix_modules' command to all spare -type gates patch netlist and select the closest spare instances for each gate in the patch netlist.
- Report ECO status and write out ECO results.
- Run the above GofCall script

2.2.4 Example GofCall script for Metal Only ECO

The GofCall script has the exact same syntax of Perl script. But it can execute exported commands that access the netlist database and modify the netlist.

The following shows an example of an automatic Metal Only ECO:

```
# GofCall ECO script, run_metal_only_example.pl
use strict;
undo_eco;# Discard previous ECO operations
# Setup ECO name
setup_eco("eco_metalonly_example " );

read_library("art.90nm.lib");# Read in standard library
read_design("-ref", "reference.gv");# Read in Reference Netlist
read_design("-imp", "implementation.gv");# Read in Implementation Netlist Which is under ECO

fix_modules("topmod");
set_top("topmod");# Set the top module that ECO is working on
   # The following is metal ECO related
   read_def("topmod.def");# Read Design Exchange Format file, optional
   # Specify spare cell pattern, when 'map_spare_cells' is done, a new spare list file will be written out
   # with updated spare list.
get_spare_cells("*/*_SPARE*");
# Comment the above line and use the following line to use spare list file
# if the spare list file has been generated already and gone through other ECOs
# get_spare_cells("-file", "spare_list_file.txt");
map_spare_cells();
# Use one of the following lines if external Synthesis Tool is used
#map_spare_cells ( "-syn", "rc" );
#map_spare_cells ( "-syn", "dc_shell" );
   report_eco(); # ECO report
   check_design("-eco");# Check if the ECO causes any issue, like floating
   write_verilog("eco_verilog.v");# Write out ECO result in Verilog
   write_soce("eco_soce.tcl");# Write out TCL script for SOC Encounter

   exit;# Exit when the ECO is done, comment it out to go to interactive mode when 'GOF >' appears
```

2.2.5 Run and debug

The GofCall Script can be run by '-run' option.

gof -run run_metal_only_example.pl

User can insert 'die' command to let GOF stop in some point and do interactive debugs when 'GOF >' shell appears. GUI mode can be enabled by run 'start_gui' command.

See Run and debug GofCall script section for more detail

2.2.6 Gated clocks in AutomaticMetal Only ECO

If the automatic metal only ECO has new gated clock cells added while the spare gates list doesn't have gated clock cell, "convert_gated_clocks" API should be run to convert gated clock cells to 'MUX' type logic. GOF will map the 'MUX' type logic to the spare type gates in 'map_spare_cells' API.

```
get_spare_cells("*/*_SPARE*");
Convert_gated_clocks();
map_spare_cells();
```

2.3 Timing ECO Flow

2.3.1 Overview

GOF can read in PrimeTime report file and do the analysis by schematic and layout placement. Cell sizing up/down, buffer insertion and cell placement adjustment can be done by GUI or script. This flow is supplement of Prime-time timing ECO flow. It is good to manually fix clock trees and last remaining violating paths after Prime-time timing ECO.

2.3.2 Files and data requirements

- Standard library (Synopsys Liberty) files with extension '.lib'.
- Other Verilog libraries if '.lib' files can't cover.
- Implementation Netlist.
- DEF file
- LEF files
- Prime-time timing report file.

2.3.3 Procedures

2.3.3.1 Read in database

Use command line to read in libraries and netlist.

```
gof -lib tsmc_std.lib -lef tsmc_std.lef -def top.def top.v
```

GofViewer Window is up after this command

2.3.3.2 Parse Primetime Report File

Use menu Commands->'Process Timing Violations' to load in Primetime report file. 'Violations Summary' windows pops up after loading file.

2.3.3.3 Check Path on LayoutViewer

Click on one path in the 'Violations Summary' window, right click to select 'Draw path on new schematic'. When a new Schematic window pops up, select portion or all the circuit on the schematic and display them on LayoutViewer.

2.3.3.4 ECO Operations

According to the analysis of the cell size, delay number and layout position, size cell or move cell ECO operations can chosen to fix the violated path.

Refer to a timing ECO example.

2.4 Script Mode Full Layers Manual ECO Flow

2.4.1 Overview

In many cases, the ECO operations are well known by users. They can be inserting buffers to a 128bits bus, or adding isolation AND gates to all outputs of a module. In these cases, manual ECO by scripts is more efficient and resource saving.

GOF exports many APIs for ECO operations in GofCall script.

2.4.2 Steps to do Manual ECO In Scripts

A typical situation for a Manual ECO:

- Run LEC on modified RTL to Implementation Netlist.
- Collect the failing points in the above run.
- Draft a GofCall script:
- Define ECO name in 'setup_eco'.
- Load Standard Cell libraries and Verilog libraries.
- Load Implementation Netlist.
- Locate ECO point.
- Use ECO APIs to fix the logic.
- Run the script.
- Report ECO status and write out ECO results.

2.4.3 Locate ECO point

Locating ECO point is the hardest part in manual ECO. Wire names in RTL codes are normally optimized away by synthesis process. GOF has a feature to retrieve the nets. Check here for the GUI way or use API ' get_match_nets' in GofCall script.

2.4.4 Files and data requirements

- Standard library (Synopsys Liberty) files with extension '.lib'.
- Other Verilog libraries.
- Implementation Netlist.
- ECO locations.

2.4.5 ECO APIs list

These APIs change Implementation Netlist

- change_pin: Modify pin connection in ECO.
- change_gate: Modify an instance type in ECO.
- change_net: Change an existing net's driver.
- change_port: Change an output port's driver.
- new_net: Create a new net.
- new_gate: Create a new gate instance.
- new_port: Create a new port.
- del_net: Delete a net.
- del_gate: Delete a gate.
- del_port: Delete a port.

For the full list of the APIs, user can type 'help' in 'GOF >' shell.

For the individual API, type 'help api_name' . For example:

```
GOF > help new_port
Help for new_port
new_port: ECO command. Create a new port for the current top level module
ECO command. Create a new port for the current top level module
Usage: new_port($name, @options);
$name: Port name
@options:
-input: New an input port
-output: New an output port
-inout: New an inout port
Note: The port name has to be pure words or with bus bit, like, abc[0], abc[1]
Examples:
new_port('prop_control_en', '-input'); # create an input port naming 'prop_control_en'
new_port('prop_state[2]', '-output'); # create an output port with bus bit 'prop_state[2]'
new_port('prop_state[3]', '-output'); # create an output port with bus bit 'prop_state[3]'
```

2.4.6 Example GofCall script for Manual ECO

```
# GofCall ECO script, run_example.pl
use strict;
undo_eco;# Discard previous ECO operations
setup_eco("eco_example");# Setup ECO name
read_library("art.90nm.lib");# Read in standard library
read_design("-ref", "reference.gv");# Read in Reference Netlist
read_design("-imp", "implementation.gv");# Read in implementation Netlist Which is under ECO
set_top("topmod");# Set the scope to the module that ECO is working on

# The following API adds a mux in flop 'state_reg_0_' D input pin,
# and connect up the original connection to pin 'A',
# pin 'B' connect to net 'next_state[7]', and pin 'S' to net 'sel_mode'
# the net can be replaced by format of 'instance/pin' , E.G. '.S(state_reg_2_/Q)'
change_pin("state_reg_0_/D", "MX2X4", "", ".A(-),.B(next_state[7]),.S0(sel_mode)");

report_eco();
write_verilog("eco_verilog.v");# Write out ECO result in Verilog
write_soce("eco_soce.tcl");# Write out TCL script for SOC Encounter

exit;# Exit when the ECO is done, comment it out to go to interactive mode when 'GOF >' appears
```

2.4.7 Run and debug

CheckRun and debug GofCall script sectionfor more detail

2.4.8 Handle repetitive work

A Perl 'for' or 'foreach' loop can handle repetitive work efficiently. For example, to add a 'AND' isolation gate for every output port of a module.

```
# GofCall ECO script, add_ands.pl
use strict;
undo_eco;# Discard previous ECO operations
setup_eco("eco_example");# Setup ECO name
read_library("art.90nm.lib");# Read in standard library
read_design("-ref", "reference.gv");# Read in Reference Netlist
read_design("-imp", "implementation.gv");# Read in implementation Netlist which is under ECO
set_top("topmod");# Set the top module that ECO is working on
my @ports = get_ports("-output");# Get all output ports of module 'topmod'
# For each output port of 'topmod', insert an 'AND' gate to enable it only when 'enable_out' is high
foreach my $port (@ports){
change_port($port, "AND2X2", "", "-,enable_out");
}

report_eco();
write_verilog("eco_verilog.v");# Write out ECO result in Verilog
write_soce("eco_soce.tcl");# Write out TCL script for SOC Encounter

gexit;# Exit when the ECO is done, comment it out to go to interactive mode when 'GOF >' appears
```

2.4.9 Special character

The special character '-' is used to represent existing connection. For example

change_pin("U0/A", "BUFFX1", "eco_buf","-");

A buffer is inserted into A pin of instance U0. The old existing net drives the new buffer now.

The special character '.' is used in ECO new instance name if the new instance needs to be in the same hierarchy as the ECO spot.

change_pin("u_qcif/u_num2/u_spare1/B", "AOI21X2",".", "net1,net2,net3");

If the instance is empty, GOF creates 'AOI21X2' in the current top level. With ".", GOF creates 'AOI21X2' new instance in hierarchy "u_qcif/u_num2/u_spare1".

2.5 Script Mode Metal Only Manual ECO Flow

2.5.1 Overview

In Manual Metal Only ECO, any new added gates will be automatically mapped to spare gate instances by 'map_spare_cells' command. A Design Exchange Format file has to be loaded for the tool to find optimal spare instances. If the file is not present, the mapping is skipped.

2.5.2 Files and data requirements

- Standard library (Synopsys Liberty) files with extension '.'lib'.
- Other Verilog libraries.
- Implementation Netlist.
- DEF (Design Exchange Format) file. If it is not loaded, GOF won't map the spare gate type cells to the exact spare instances.
- Spare gates pattern. It is in 'hierarchical_instance/leaf_instance' format. It has wild card '*' to match the spare gates in Implementation Netlist.
- Spare gates list file. If several users work on the same Implementation Netlist, the initial spare gates list file should be generated only once. And a new spare gates list file should be created every time ECO is done.
- ECO locations.

2.5.3 Example GofCall script for Manual Metal Only ECO

```
# Manual Metal Only ECO, manual_metal_eco.pl
use strict;
undo_eco;
setup_eco("metal_eco0123");
set_log_file("metal_eco0123.log");
read_library("/prj/lib/tsmc40.lib");
read_design("-imp", "/prj/netlist/imp_net.v");
set_top("mtop");

new_port("nout7", "-output");# Create a new port 'nout7'
# Place the port to 60000, 1000000. It's approximate position, the main purpose is for
# spare instances selection
place_port("nout7", 60000, 100000);
new_port("nout8", "-output");# Create another port
place_port("nout8", 120000, 81000);
# 'nout8' is driven by an invert first, and the invert's input is driven by pin 'cmpmod/rego/QN'
change_port("nout8", "INV_X1M", "", "cmpmod/rego/QN");
# Drive the 'nout7' by 'INV_X1M' and leave the input unconnected, but the mapped
# spare instance name is returned.
my $inst = change_port("nout7", "INV_X1M", "", "");
# Drive the new instance's input by a flop, and specify the flop's connection in the 4thargument
change_pin("$inst/A", "SDFFRPQ_X4M", "", \
".CK(cmpmod/rego/CK),.D(cmpmod/rego/QN),.R(1'b0),.SE(1'b0),.SI(1'b0)");

read_def("/prj/def/imp_net.def");
get_spare_cells("Spare_*/*_SPARE_GATE*");
# Before mapping to spare gates, set a large number in buffer distance, so that GOF will not
# add buffers for long connections.
set_buffer_distance(9999999);
# The following 'map_spare_cells' command maps the three new ECO instances to the optimal
# spare instances.
map_spare_cells;

report_eco;
write_verilog("imp_eco0123.v");
```

2.5.4 Run and debug

The GofCall Script can be run by '-run' option.

gof -runmanual_metal_eco.pl

Check Run and debug GofCall script sectionfor more detail

2.6 GUI Mode Full Layers ECO Flow

2.6.1 Overview

The following paragraph demonstrates how to insert buffers and inverters into a circuit in GUI mode.

2.6.2 Start up GOF in GUI Mode

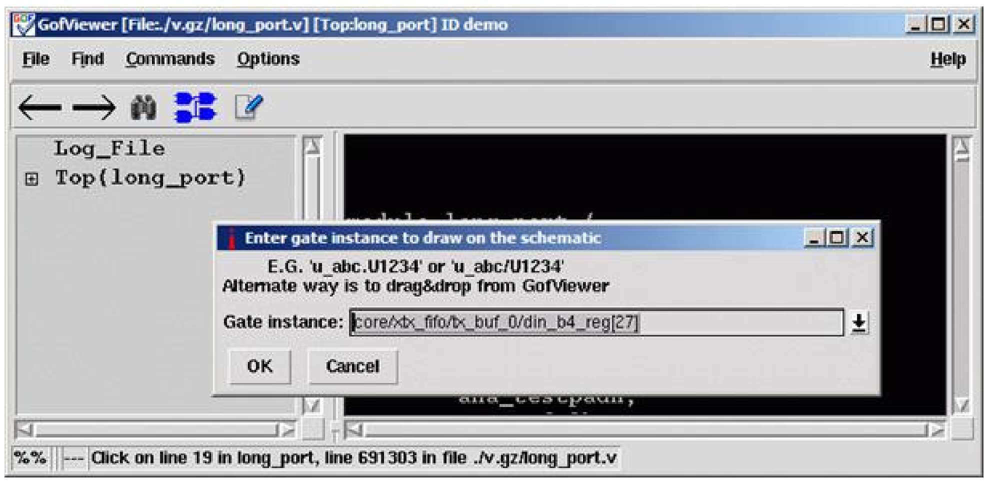
Start up Gates On the Fly by the command line

gof -lib t65nm.lib -lib io.lib netlist_port.v

For detail usage, visit this link

http://www.nandigits.com/usage.htm

In GofViewer netlist window, press ctrl-g or menu commands->'Launch GofTrace with gate'. Fill in the instance name that needs ECO.


GofViewer [File:./v.gz/long_port.v] [Top:long_port] ID demo
File
Find
Commands
Options
Help
Log_File
Top{long_port}
Enter gate instance to draw on the schematic
OK
Cancel
%%

E.G. 'u_abc.U1234' or 'u_abc/U1234'
Alternate way is to drag&drop from GofViewer

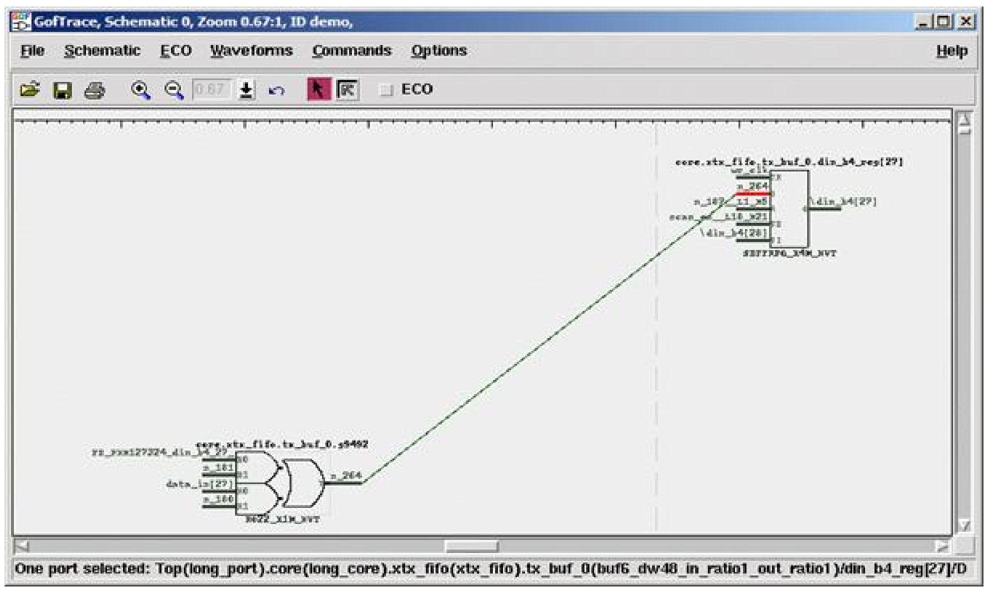
Gate instance: core/xtx_fifo/tx_buf_0/din_b4_reg[27]

--- Click on line 19 in long_port, line 691303 in file ./v.gz/long_port.v

Figure 7 Load gate to schematic

2.6.3 Create Partial Schematic

In GofTrace schematic window, use mouse middle button to expand the schematic. In this case, pin D of the flop should be inserted an invert.

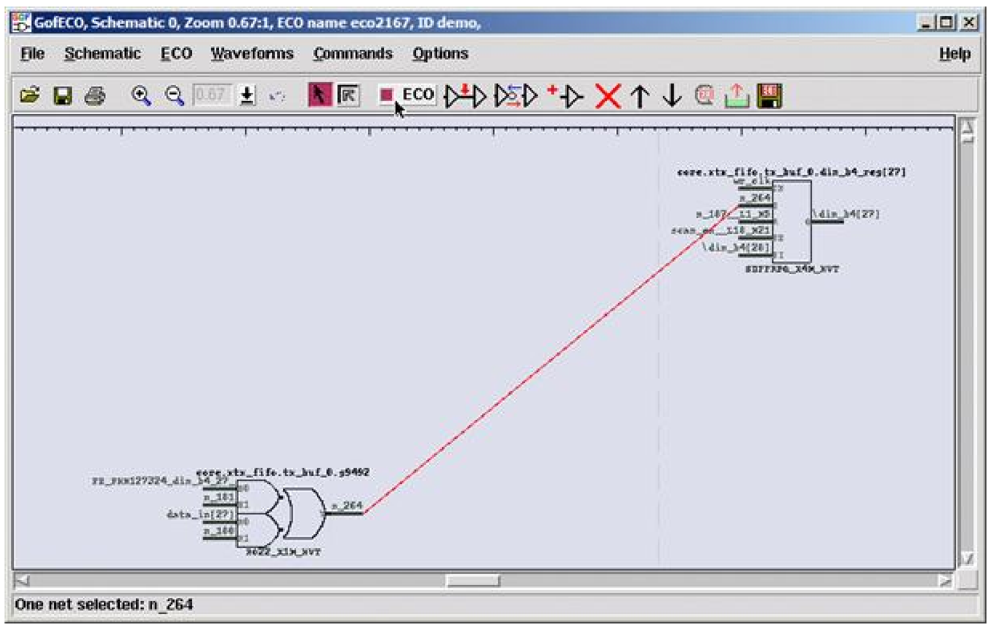

GofTrace, Schematic 0, Zoom 0.67:1, ID demo,

One port selected: Top(long_port).core(long_core).xtx_fifo(xtx_fifo).tx_buf_0(buf6_dw48_in_ratio1_out_ratio1)/din_b4_reg[27]/D

Figure 8 Partial Schematic for GUI ECO

2.6.4 Do ECO on schematic

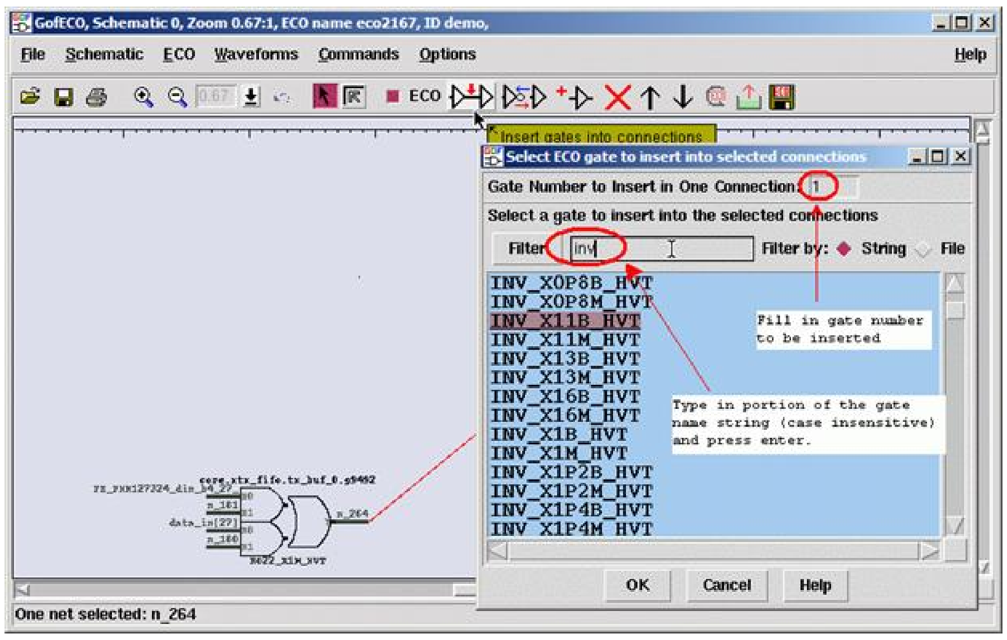
Check ECO button to enable ECO mode


GofECO, Schematic 0, Zoom 0.67:1, ECO name eco2167, ID demo,

One net selected: n_264

Figure 9 Schematic in ECO Mode

Press mouse-left-button on the wire to select it. Click ECO button 'Insert gates into connections', select the right invert in the gate type selection window.


GofECO, Schematic 0, Zoom 0.67:1, ECO name eco2167, ID demo,

Select ECO gate to insert into selected connections

Gate Number to Insert in One Connection: 1

Select a gate to insert into the selected connections

Filter inv Filter by: ● String ○ File

INV_X0P8B_HVT
INV_X0P8M_HVT
INV_X11B_HVT
INV_X11M_HVT
INV_X13B_HVT
INV_X13M_HVT
INV_X16B_HVT
INV_X16M_HVT
INV_X1B_HVT
INV_X1M_HVT
INV_X1P2B_HVT
INV_X1P2M_HVT
INV_X1P4B_HVT
INV_X1P4M_HVT

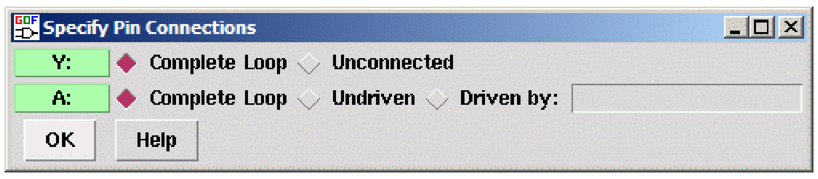
Fill in gate number to be inserted

Type in portion of the gate name string (case insensitive) and press enter.

One net selected: n_264

Figure 10 Select Gate in GUI ECO

In 'Pin Connections' setup window, use default 'Complete Loop' option, so that the gate can be inserted in the net.

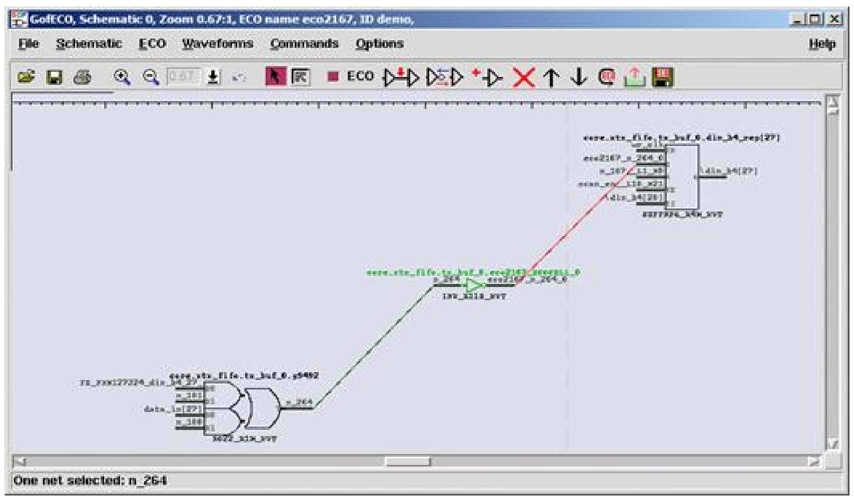

Specify Pin Connections

Y: ● Complete Loop ○ Unconnected

A: ● Complete Loop ○ Undriven ○ Driven by:

Figure 11 New Cell Pin Connection Selections

Click OK and the invert is inserted.

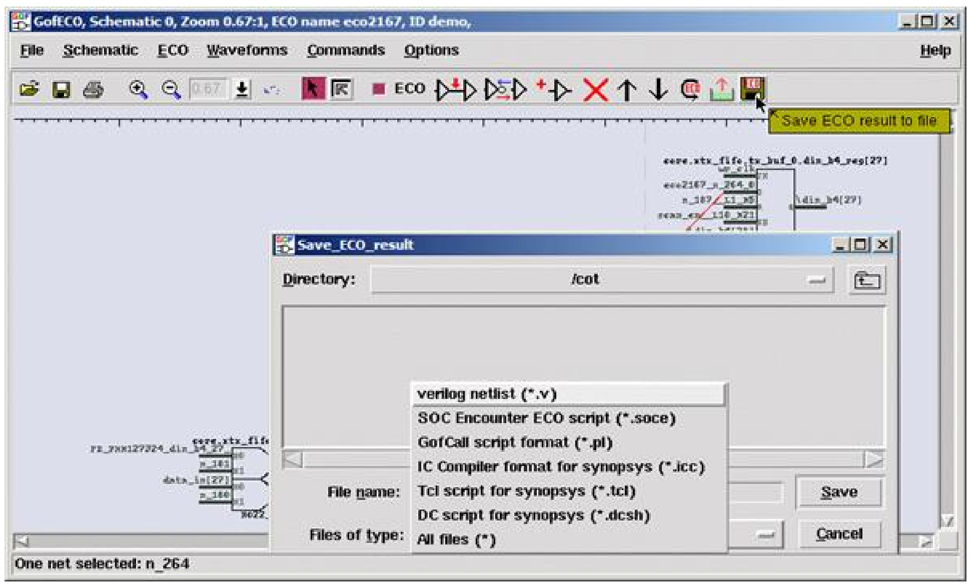

GofECO, Schematic 0, Zoom 0.67:1, ECO name eco2167, ID demo,
File
Schematic
ECO
Waveforms
Commands
Options
Help
ECO
core.xtx_fifo.tx_buf_0.din_b4_reg[27]
eco2167_n_264_0
din_b4[27]
n_264
INV_X1
data_in[27]

One net selected: n_264

Figure 12 Manual ECO with New Gate Inserted

2.6.5 Save ECO

Press ECO button ‘Save ECO result to file’. And select the format to be saved. The supported formats include verilog netlist, SOC Encounter ECO script, GofCall Script, TCL script and DCShell script.


GofECO, Schematic 0, Zoom 0.67:1, ECO name eco2167, ID demo,
File
Schematic
ECO
Waveforms
Commands
Options
Help
ECO
Save ECO result to file
core.xtx_fifo.tx_buf_0.din_b4_reg[27]
Save_ECO_result
File name:
Files of type:
verilog netlist (*.v)
SOC Encounter ECO script (*.soce)
GofCall script format (*.pl)
IC Compiler format for synopsys (*.icc)
Tcl script for synopsys (*.tcl)
DC script for synopsys (*.dcsh)
All files (*)
Save
Cancel

Directory: /cot

One net selected: n_264

Figure 13 Save ECO in GUI Mode

2.7 GUI Mode Metal Only ECO Flow

2.7.1 Overview

Metal ECO can only use existing spare gates on the silicon. Gates On the Fly controls how to use these spare gates.

2.7.2 Methods for Metal Only ECO

Four methods are supported in Metal Only ECO:

·User can add any type of gates and let the tool map to the spare type gates, Place and Route tool should map the spare type gates to the exact spare gate instances.

·User can add any type of gates and let the tool map to the exact spare gate instances.

·User can add only spare type gates and let the tool map to the exact spare gate instances.

·User can pick the exact spare gate instances, and connect and disconnect up the instances in ECO.

Note: 'Spare type gate' refers to the gate type, 'INVX2', 'NAND2X2'. 'Exact spare gate instance' refers to the spare instances in the design, E.G. 'spare1/spare_invx2'

2.7.3 Setup and use cases

The detail setup for four method can be found inGofECO Metal Only ECO. Use cases can be found inonline document.

3 Guidance to choose Netlist ECO flows

3.1 Overview

ECO cases vary in complexity. It can be as simple as one cut or adding one gate and it can be as complicated as involving thousands gates.

3.2 Decision Factors

- Timing or functional

  Timing ECO up-sizes or down-sizes cells, buffers or inverters can be added or deleted. It doesn't change netlist functionality.

  Functional ECO involves function changes.

- Metal only or full layers

  Metal only ECO changes metal layers, any new gates should use spare gates.

  Full layers ECO can modify netlist freely.

- Close to flops or ports or instances with name preserved

  When ECO spot is close to flops or ports or instances with name preserved, wire and instance names will be kept the same as the RTL. Manually modification becomes possible.

  The change in RTL is on some combinational wires which may have been optimized away or may have different functionality in netlist even with the same name. Automatic ECO with reference netlist is recommended in this case.

- Random or Repeated

  Random ECO can be in any location of the netlist. Multiple spots in the netlist may have to be modified with different operations.

  Repeated ECO does same operations on a group of wires/ports/instances, for example, adding AND gates to all output ports for one module. It is more efficient to do these ECOs in manual script mode.

- Involving more than 50 gates

  ECO involving more than 50 gates becomes hard for manually modification and if the change is not repeated and not close to flops and ports, automatic should be chosen. However, this threshold number depends on users, it can be increased or decreased.

  When ECO has less than the threshold gate number and the ECO spot is close to flops and ports, manually modification becomes feasible.

- GUI or script mode

  GUI mode is easier to ramp up. For users not doing ECO frequently, GUI mode is a good choice.

  Script mode is very powerful and efficient, and it's reproducible. When the ECO is repeated operations, script mode is preferred.

3.3 ECO flows decision tree

The ECO flows decision tree is illustrated in the diagram below:

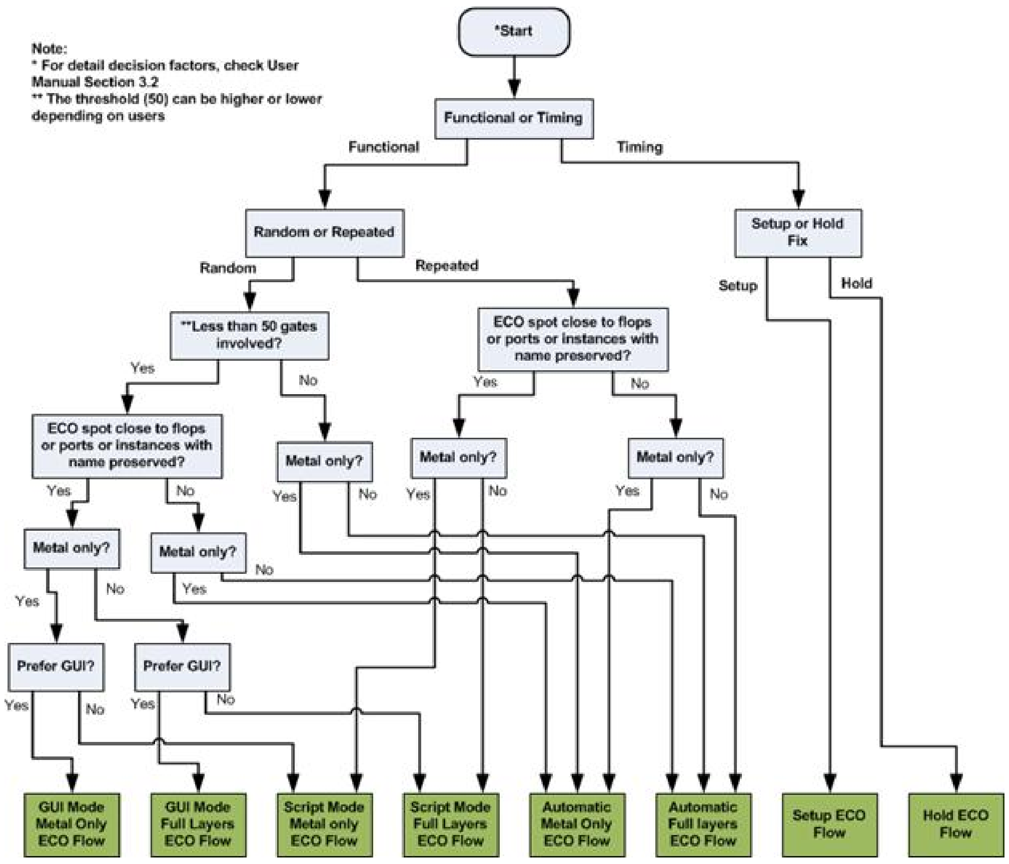

Note:
* For detail decision factors, check User Manual Section 3.2
** The threshold (50) can be higher or lower depending on users
*Start
Functional or Timing
Functional
Timing
Random or Repeated
Random
Repeated
Setup or Hold Fix
Setup
Hold
**Less than 50 gates involved?
ECO spot close to flops or ports or instances with name preserved?
Yes
No
Metal only?
Prefer GUI?
GUI Mode Metal Only ECO Flow
GUI Mode Full Layers ECO Flow
Script Mode Metal only ECO Flow
Script Mode Full Layers ECO Flow
Automatic Metal Only ECO Flow
Automatic Full layers ECO Flow
Setup ECO Flow
Hold ECO Flow

Figure 14 ECO Flow Decision Tree

4 Script Mode Detail Features

4.1 GofCall Script

GofCall is the Perl Script Interface which can access internal exported APIs.

4.1.1 Get Help

- Type 'help' in interactive shell 'GOF >' to list all APIs.
- Type 'help set_*' to list all APIs matching 'set_', like 'set_top', 'set_invert'
- Type 'help individual-API' to list detail of the API.
- Visitwww.nandigits.com/gof_manual.htmfor online manual.

4.1.2 Feature list

- Compatible with Perl.
- Automatic ECO with fix_modules/fix_logic/fix_setup/fix_hold APIs.
- Rich ECO APIs to do manual ECO.
- ECO operations are reversible.
- ECO result can be loaded in schematic on the fly.
- Integrated commands to browser netlist and check design status

4.1.3 API list

buffer: ECO command. Buffer high fanout ECO nets
change_gate: ECO command. Modify an instance in ECO.
change_net: ECO command. Change a existing net's driver
change_pin: ECO command. Modify pin connection in ECO.
change_port: ECO command. Change an output port's driver
check_design: Check if the design has unresolved modules
compare: Logic equivalence check on output port and register input pins
compare_nets: Check equivalence of two nets in the reference and implementation netlist
convert_gated_clocks: ECO command. Convert gated clocks to MUX logic
current_design: Set the current top level module
current_instance: Set the current instance
del_gate: ECO command. Delete gate
del_net: ECO command. Delete net
del_port: ECO command. Delete port
exist_inst: Check if an instance exists
exist_wire: Check if exist the wire
fix_hold: ECO command. Fix hold time violations.
fix_logic: ECO command. Fix listed points
fix_modules: ECO command. Fix modules listed
fix_setup: ECO command. Fix Setup time violations.
flatten_modules: Flatten hierarchical modules in reference netlist
get_cell_info: Get information of a module or instance
get_cells: Get all cells in the current module or sub-modules
get_conns: Get connections of net or pin in the top level module
get_coord: Get an instance's coordination
get_definition: Get instantiation of instance
get_driver: Get the driver of a net or pin
get_drivers: Get the drivers of a net or pin
get_instance: Get instance in the top level module
get_instances: Get all hierarchical instances in the top level module
get_leaf_pin_dir: Get leaf cell pin's direction input/output/inout
get_leafs_count: Get all leaf cells name and count in the top level module
get_lib_cells: Get leaf gates in libraries
get_loads: Get loads of net or pin in the top level module
get_logic_cone: Get logic cone of nets or pins
get_match_nets: Get logic equivalent nets in implementation netlist
get_modules: Get all hierarchical modules under current module
get_net_of: Get the net name connecting to the pin
get_nets: Get nets that matching pattern
get_path: Get current hierarchical path
get_pins: Get pins of instance or module
get_ports: Get all ports in the current top level module
get_ref: Get the reference of the instance
get_resolved: Resolve the relative path to module and leaf item
get_roots: Get root designs name
get_spare_cells: ECO command. Get spare cells
gexit: Exit the GofCall interactive mode
gprint: Print out the message and save to log file
is_leaf: Check if a module or instance is leaf cell
is_seq: Check if a leaf module is a specific sequential cell

lv_search: Search instances in LayoutViewer window
map_spare_cells: ECO command. Map all new created cells to spare cells
new_gate: ECO command. Create new gate
new_net: ECO command. Create a new net
new_port: ECO command. Create a new port for the current top level module
place_gate: ECO command. Place gate position
place_port: ECO command. Place port position
pop_top: Pop out the saved top level module from the stack and discard the current setting
push_top: Set the current top level module and push the previous setting to stack
read_def: Read DEF file
read_design: Read verilog netlist
read_file: Read timing violation report file
read_lef: Read LEF file
read_library: Read standard library or verilog library files
rename_net: ECO command. Rename a net name
report_eco: Report ECO
report_spares: Report Spare cells
run: Run GofCall script
sch: Launch schematic to verify ECO
set_bound_opti: Enable or disable boundary optimization check. Enabled by default
set_buffer: Set buffer type. The tool will automatically pick one if the command is not called
set_buffer_distance: Set distance limit for inserting buffer
set_constraints: Set contraints for map_spare_cells command.
set_dont_use: Set dont use list.
set_exit_on_error: Whether the tool should exit when the script runs into an error
set_exit_on_warning: Whether the tool should exit when the script runs into a warning
set_ignore_network: Set ignore network
set_inst: Set the current instance
set_invert: Set invert type. The tool will automatically pick one if the command is not called
set_keep_format: Set keep format of the original verilog when ECO is done
set_leaf: Set a hierarchical module to be leaf. Useful to stub hierarchical instances.
set_log_file: Set log file name
set_max_lines: Set max output lines
set_max_loop: Setup max loop
set_mod2mod: Set reference module mapping to implementation module
set_mu: MU configuration
set_power: Set power pins connections for leaf cell
set_preserve: Set preserve property on instances. The tool will not remove them in ECO
set_quiet: Run script in quiet mode
set_recovery_distance: Set distance limit for gates recovery in ECO
set_tiehi_net: Set tiehi net name
set_tielo_net: Set tielo net name
set_top: Set the current top level module
set_tree: Set the current tree
set_verbose: Run script in verbose mode
setup_eco: ECO command. Setup ECO
start_gui: Start GUI windows
stitch_scan_chain: ECO command. Stitch scan chain
suppress_warnings: Suppress warning messages
swap_inst: ECO command. Swap two instances with same input/output pins.
undo_eco: ECO command. Undo eco operations
write_dcsh: ECO command. Write out ECO result in Design Compiler dcsh script format
write_perl: ECO command. Write out GofCall ECO script compatible with Perl.
write_soce: ECO command. Write out ECO result in Cadence SOC Encounter script format
write_spare_file: ECO command. Write out spare cells list file
write_tcl: ECO command. Write out ECO result in Design Compiler tcl script format
write_verilog: ECO command. Write out ECOed netlist to a file

4.1.4 API sgrouping

4.1.4.1 Netlist Browse APIs

One key element to do efficient manual ECO is to isolate the ECO spots quickly. The following APIs are for fast Netlist Browsing.

get_cells: Get all cells in the current module or sub-modules
get_conns: Get connections of net or pin in the top level module
get_driver: Get the driver of a net or pin
get_drivers: Get the drivers of a net or pin
get_instance: Get instance in the top level module
get_instances: Get all hierarchical instances in the top level module
get_leaf_pin_dir: Get leaf cell pin's direction input/output/inout
get_lib_cells: Get leaf gates in libraries
get_loads: Get loads of net or pin in the top level module
get_match_nets: Get logic equivalent nets in implementation netlist
get_modules: Get all hierarchical modules under current module
get_net_of: Get the net name connecting to the pin
get_nets: Get nets that matching pattern
get_pins: Get pins of instance or module
get_ports: Get all ports in the current top level module
get_ref: Get the reference of the instance

For example, to get all flops' data pins in one module. The script can use these browse APIs

```
my @flop_data_pins;
set_top("module_name");
my @flops = get_cells("-type", "ff");
foreach my $flop (@flops){
  my @dpins = get_pins("-data", $flop);
  push @flop_data_pins, @dpins;
}
```

After the script is run, @flop_data_pins have all data pins of all flops in the module.

4.1.4.2 Automatic ECO APIs

These APIs are for Automatic ECO

fix_modules: ECO command. Fix modules listed
map_spare_cells: ECO command. Map all new created cells to spare cells
fix_logic: ECO command. Fix listed points

Combining netlist browsing APIs, users can come up a short script to do complicated changes.

For example, to fix all modules named "tx_machine_*"

```
my @modules = get_modules("-hier", "tx_machine_*");
fix_modules(@modules);
```

4.1.4.3 File IO APIs

These APIs are for reading/writing files.

read_def: Read DEF file
read_design: Read verilog netlist
read_file: Read timing violation report file
read_lef: Read LEF file
read_library: Read standard library or verilog library files
write_dcsh: ECO command. Write out ECO result in Design Compiler dcsh script format
write_perl: ECO command. Write out GofCall ECO script compatible with Perl.
write_soce: ECO command. Write out ECO result in Cadence SOC Encounter script format
write_spare_file: ECO command. Write out spare cells list file
write_tcl: ECO command. Write out ECO result in Design Compiler tcl script format
write_verilog: ECO command. Write out ECOed netlist to a file

4.1.4.4 Manual ECO APIs

These are APIs for Manual ECO.

buffer: ECO command. Buffer high fanout ECO nets
change_gate: ECO command. Modify an instance in ECO.
change_net: ECO command. Change a existing net's driver
change_pin: ECO command. Modify pin connection in ECO.
change_port: ECO command. Change an output port's driver
del_gate: ECO command. Delete gate
del_net: ECO command. Delete net
del_port: ECO command. Delete port

new_gate: ECO command. Create new gate
new_net: ECO command. Create a new net
new_port: ECO command. Create a new port for the current top level module

Combining netlist browsing APIs, a short GofCall script can do very efficient ECOs.

For example, to add isolation cells for all output ports of a module.

```
set_top("module_name");
my @out_ports = get_ports("-output");
foreach my $out (@out_ports){
  change_port($out, "AND2X2", "", "-,net_iso_enable");
}
```

4.1.5 API usage

For detail of APIs visithttp://www.nandigits.com/gofcall_apis.htm

For individual API, type '#' followed by API_name.

For example, to check usage of 'fix_modules'

http://www.nandigits.com/gofcall_apis.htm#fix_modules

4.2 String Handling In Script Mode

4.2.1 Single quote and double quote

Any string in GofCall script for module/instance/wire/pin/port should be enclosed by single quote or double quote. When a Perl variable is used, double quote should be used

```
my $inst = "state_reg_0";
change_pin("$inst/D","1'b0");
```

4.2.2 Instance and netwith backslash

Instance with backslash should be either put in single quote and with a space in the end, or removed the backslash and let the tool to figure out the real name.

'\u_abc/u_def/state_reg[0] ' can be written as 'u_abc/u_def/reg_state_0[0]'

Net name with backslash should always keep the backslash. One good way to avoid write net name directly is to use 'instance/pin' format. For example

DFFQ_X4M \u_abc/u_def/state_reg[0] (.D(\u_abc/u_def/net123), .Q(\u_abc/u_def/state[0]));

The net '\u_abc/u_def/net123 ' can be replaced by 'u_abc/u_def/state_reg[0]/D' and

The net '\u_abc/u_def/state[0] ' can be replaced by 'u_abc/u_def/state_reg[0]/Q'

4.3 Run and debug GofCall script

4.3.1 Command line

In command line, the GofCall Script can be run by '-run' option.

gof -run run_example.pl

4.3.2 GOF Shell

If '-run' option is present in the command line, and 'gexit' is not in the script, or GOF meets error when executing the script, GOF goes to interactive mode with GOF shell 'GOF >'.

```
GofCall, Netlist Processing Script APIs, Interactive Mode
Run 'start_gui' to launch GUI window
Run 'help' to list API calls
GOF >
```

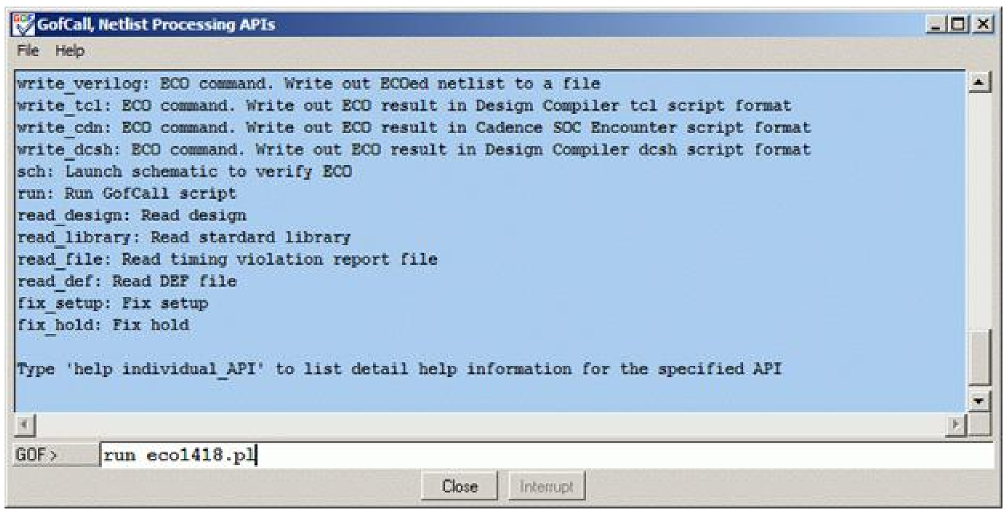
Individual command can be executed in GOF shell. The command can be in nested mode

GOF > set_top(get_ref("u_rxbuf"))

4.3.3 Run in GUI mode

GofCall scripts can be run in GUI window. In GofViewer, click Menu Commands->'Launch GofCall Script Interface' to launch GofCall GUI window.

Type 'help' in the shell entry for help information. Scripts can be run by 'run' command in the shell entry



```
write_verilog: ECO command. Write out ECOed netlist to a file
write_tcl: ECO command. Write out ECO result in Design Compiler tcl script format
write_cdn: ECO command. Write out ECO result in Cadence SOC Encounter script format
write_dcsh: ECO command. Write out ECO result in Design Compiler dcsh script format
sch: Launch schematic to verify ECO
run: Run GofCall script
read_design: Read design
read_library: Read stardard library
read_file: Read timing violation report file
read_def: Read DEF file
fix_setup: Fix setup
fix_hold: Fix hold

Type 'help individual_API' to list detail help information for the specified API

GOF > run eco1418.pl
```

Figure 15 GofCall window

4.3.4 Fast schematic launch

In GOF shell, GUI windows can be launched by 'start_gui' or 'sch' commands.

'start_gui' launches netlist view window first and user can bring up schematic window from netlist view window.

'sch' command only launches schematic window, and it doesn't enable netlist view window. So it has fast turnaround in GUI interactive debug.

For example,

After the following command is done,

change_pin("u_top/u_core/u_regmod/state_reg/D", "XOR2X2", "", "-,new_enable");

Run 'sch' in 'GOF >'

GOF > sch("u_top/u_core/u_regmod/state_reg")

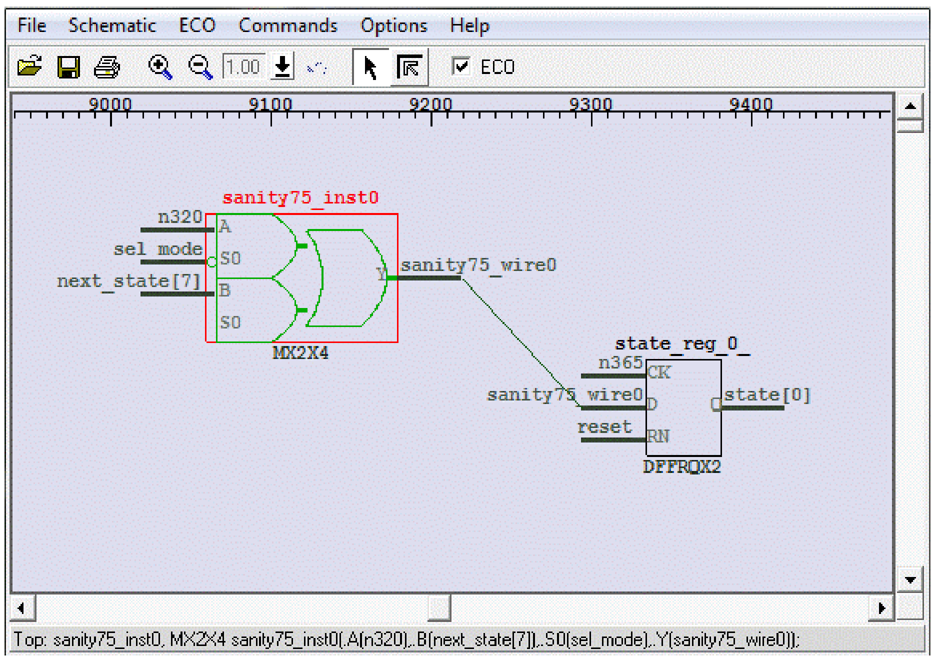
The instance will be loaded into a schematic and user can click on the instance's pins to trace fanin/fanout on the schematic to see if the ECO is done as expected.

4.3.5 Break points for debug

'sch' fast schematic launch command can be used as break points for debug. For example, 'sch' commands are inserted in GofCall script, when the tool runs to the point, a schematic is launched.

```
...
setup_eco("sanity75");
set_log_file("t_sanity75.log");
read_library("libdr/art.m.simple.lib");
read_design('-imp', "cdir/imp_name.v");
change_pin("state_reg_0_/D", "MX2X4", "", ".A(-),.B(next_state[7]),.S0(sel_mode)");
sch("state_reg_0_", "-set", 1);
...
```

On the schematic, user can use mouse-middle-button clicking on the pin 'D' to see if the ECO is done as expected.

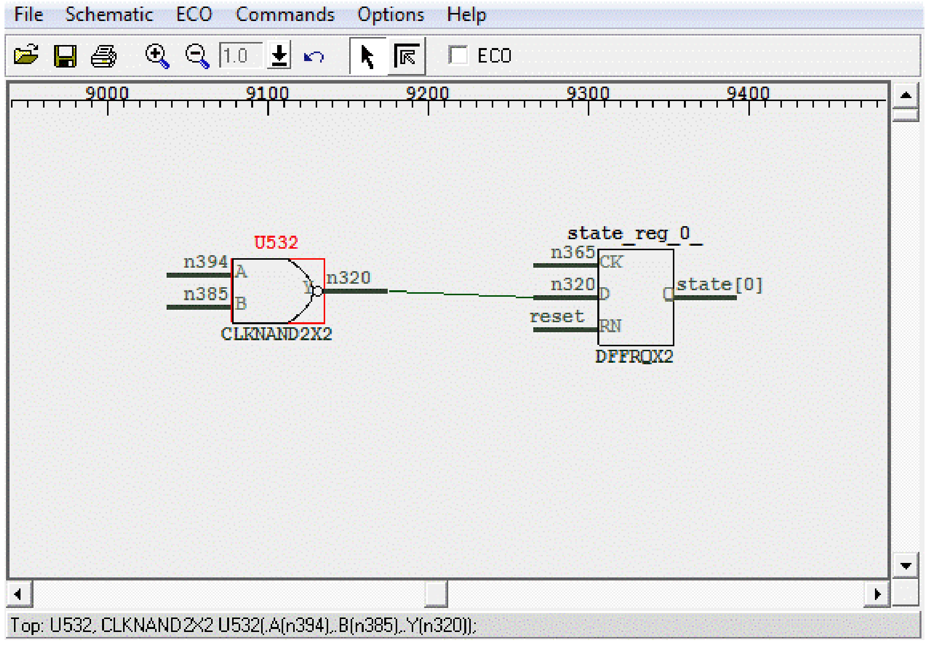


Top: sanity75_inst0, MX2X4 sanity75_inst0(.A(n320),.B(next_state[7]),.S0(sel_mode),.Y(sanity75_wire0));

Figure 16 Launch schematic at break point

Note: 'ECO' check-button is enabled automatically, since there is ECO having been done.

To compare with the logic before ECO, launch a new schematic by menu Schematic->'New Schematic'. On the new schematic, press 'ctrl-g' or by menu Schematic->'Load Gate' to load in the flop under ECO.



Top: U532, CLKNAND2X2 U532(.A(n394),.B(n385),.Y(n320));

Figure 17 Launch schematic before ECO

Note: 'ECO' check-button is un-checked.

4.4 Typical Manual ECO operations

4.4.1 Insert gate to port

Both input port and output port have the same operation

4.4.1.1 Insert an invert to input port

```
# Insert to input port 'in_enable'
change_port("in_enable", "INVX1", "inst_name", "-"); # 'inst_name' can be empty
```

4.4.1.2 Insert to output port

```
# Insert AND2X1 to output port 'out_enable', one pin connects to the original driver,
# the other pin is driven by 'scan_mode'
change_port("out_enable", "AND2X1", "", "scan_mode,-");
```

4.4.1.3 Insert inverts to multiple ports

```
# Find all ports matching string "abcde" and insert invert to each port
my @ports = get_ports("*abcde*");
foreach my port (@ports){
change_port($port, "INVX1", "", "-");
}
```

4.4.2 Insert gate to register instance pin

4.4.2.1 Insert invert to flop data pin

```
# Insert an invert to 'D' pin of flop 'abc_reg'
change_pin("abc_reg/D", "INVX1", "", "-");
```

4.4.2.2 Insert invert to flop output pin

```
# Insert an invert to 'Q' pin of flop 'abc_reg'
change_pin("abc_reg'Q", "INVX1", "", "-");
```

4.4.2.3 Insert MUX to data pin of multiple flops

```
# Find all flops matching string "cnt_reg" and insert MUX to 'D' pin of each flop,
# so that each flop is preset to 'preset_val' when 'preset_enable' is high
my @flops = get_cells("*cnt_reg*");
foreach my $flop (@flops){
change_pin("$flop/D", "MUX2X2", "", ".A(-),.B(preset_val),.S(preset_enable)");
}
```

4.4.3 Change flops to other type

4.4.3.1 Changenon-resetflop type to resettable flop

```
# Find all flops matching string "cnt_reg" and change each flop to resettable flop
my @flops = get_cells("*cnt_reg*");
foreach my $flop (@flops){
  change_gate($flop, "DFFRQX2", ".RD(reset_n)");
}
```

4.4.4 Insert gate to hierarchical instance pin

4.4.4.1 Insert invertsto hierarchical instance pins

```
# Find all instances matching "tx_mac" in module "abc_mod" and insert invert to 'loop_en' pin
set_top("abc_mod");
my @insts = get_instances("*tx_mac*");
foreach my $inst (@insts){
  change_pin("$inst/loop_en", "INVX1", "", "-");
}
```

4.4.4.2 Insert AND to hierarchical instance pins

```
# Find all instances matching "tx_mac" in module "abc_mod" and
# AND all output pins with "power_on" signal.
set_top("abc_mod");
my @insts = get_instances("*tx_mac*");
foreach my $inst (@insts){
  my @pins = get_pins("-output", $inst);
  foreach my $pin (@pins){
    my $net = get_net_of($pin); # Only add AND to those out pins driving nets
    if($net){
      change_pin($pin, "AND2X2", "", ".A(-),.B(power_on)");
    }
  }
}
```

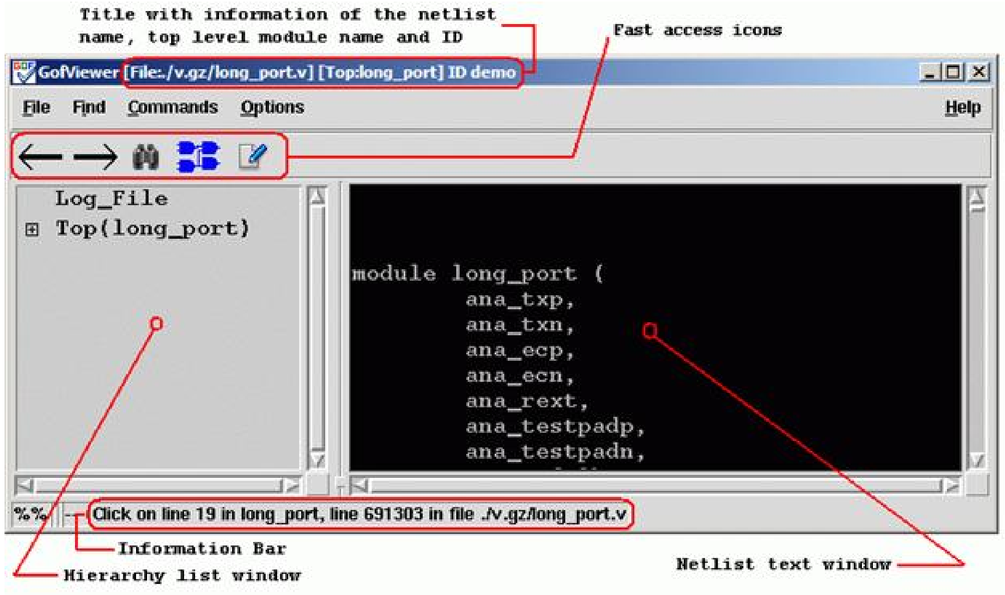
5 GUI ModeDetail Features

5.1 GofViewer

When GOF is run without '-run' and '-shell' option, it goes into GUI mode.

gof -lib t65nm.lib -lib io.liblong_port.v

GofViewer is the first window after GOF starts up GUI interface.


Title with information of the netlist
name, top level module name and ID
Fast access icons
GofViewer [File:./v.gz/long_port.v] [Top:long_port] ID demo
File
Find
Commands
Options
Help
Log_File
Top(long_port)
%%
Information Bar
Hierarchy list window
Netlist text window

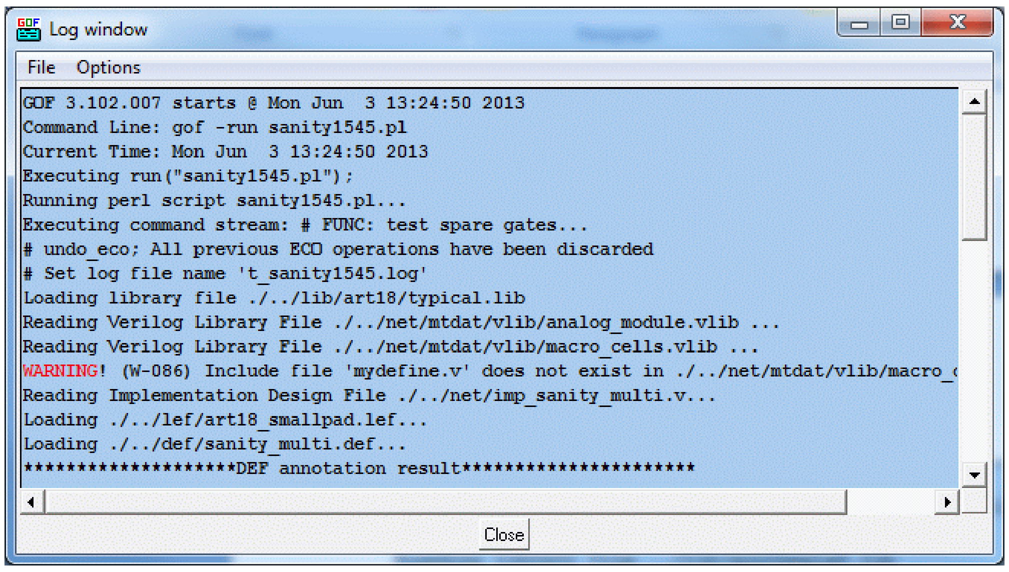
```
module long_port (
        ana_txp,
        ana_txn,
        ana_ecp,
        ana_ecn,
        ana_rext,
        ana_testpadp,
        ana_testpadn,
```

Click on line 19 in long_port, line 691303 in file ./v.gz/long_port.v

Figure 18 GofViewer Window

5.1.1 Log Window

If there errors or warnings in loading the database, Log Window pops up



```
GOF 3.102.007 starts @ Mon Jun  3 13:24:50 2013
Command Line: gof -run sanity1545.pl
Current Time: Mon Jun  3 13:24:50 2013
Executing run("sanity1545.pl");
Running perl script sanity1545.pl...
Executing command stream: # FUNC: test spare gates...
# undo_eco; All previous ECO operations have been discarded
# Set log file name 't_sanity1545.log'
Loading library file ./../lib/art18/typical.lib
Reading Verilog Library File ./../net/mtdat/vlib/analog_module.vlib ...
Reading Verilog Library File ./../net/mtdat/vlib/macro_cells.vlib ...
WARNING! (W-086) Include file 'mydefine.v' does not exist in ./../net/mtdat/vlib/macro_c
Reading Implementation Design File ./../net/imp_sanity_multi.v...
Loading ./../lef/art18_smallpad.lef...
Loading ./../def/sanity_multi.def...
*******************DEF annotation result*********************
```

Figure 19 Log Window

5.1.2 File Menu

5.1.2.1 Load Design

Users can input netlist files and design files through the Load Design command.

5.1.2.2 Reload Design

If any netlist file or design file has been updated during GOF session, this command can be used to reload the design.

5.1.2.3 Save Startup File

This command saves the inputs users have typed in Load Design window to startup file. In Windows platform, it’s a batch file with ‘bat’ extension. In Linux platform, it’s a shell file. Users can load the design by running the startup file in the next time.

5.1.2.4 Open Other Netlist

This command loads another netlist file to create a new hierarchical tree. The hierarchy tree will be listed in the hierarchy list window. The command is useful when users want to draw circuits from different netlist file on the same schematic which is good for logic comparison in netlist debug scenario such as LEC failures analysis.

5.1.2.5 Open Log Window

The command opens log file in a text window.

5.1.2.6 Exit

Exit command.

5.1.3 Find Menu

5.1.3.1 Search

This command searches for the matching string in the netlist text window.

5.1.3.2 Goto Line Number

GOF loads only one module in the netlist text window when the netlist file is hierarchical with multiple modules. The command will load in the corresponding module into the text window and highlight the line with the specific number in the netlist file.

5.1.3.3 ReportArea

This command reports the design area. The command requires standard library files to be loaded which include leaf cell area information.

5.1.3.4 Report Leakage

This command reports the leakage power in the design. Same as the Report Area command it requires standard libraries.

5.1.3.5 Report Leaf Cells

This command reports the leaf cell type and numbers in the design.

5.1.3.6 Report Submodules

This command reports the hierarchical sub-modules in the design.

5.1.3.7 Statistic of Current Design

This command reports the statistic of the current design. It pops up an option window for interactivity from users.

5.1.3.8 List Library

The command lists the libraries and leaf cells in each library.

5.1.3.9 List Context for Leaf Cell

This command pops up an entry window for users to input leaf cell name string, wild card can be accepted. All leaf cells matching the string will be listed. If there is only one cell matched, the detail property will be listed.

5.1.4 Commands Menu

5.1.4.1 Launch GofTrace Schematic

This command launches GofTrace Schematic, if any instance or net string is highlighted in the netlist window, the instance or the driver of the net will be drawn on the schematic. Otherwise, the schematic is empty.

5.1.4.2 Launch GofTrace with Gate

This command pops up an entry window for users to input a string to load a specific instance. For example, ‘u_abc/U123’. Click ‘OK’ button on the pop window, GofTrace Schematic is launched.

5.1.4.3 Launch Layout Viewer

This command launches Layout Viewer window, if any instance or net string is highlighted in the netlist window, the instance or the driver of the net will be highlighted on the Layout Viewer window. The command requires that physical files to be loaded. Both def and lef files should be loaded before launching Layout Viewer, otherwise a warning window pops up for the missing physical files.

5.1.4.4 Launch GofCall Script Interface

This command launches GofCall window to run scripts or other interactive command.

5.1.4.5 Switch to GOF Shell Mode

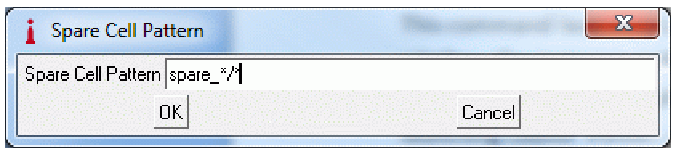
This command exits GUI mode and switches to shell mode. It has ‘GOF >’ shell interface in shell mode.

5.1.4.6 Spare Cells

This command group processes Spare cells in metal ECO. Warning! GUI metal ECO is used for visually checking the possibility of metal ECO. The script mode metal ECO is recommended.

- Create Spare Cells File

  This command extracts spare cells from netlist file. A pop window appears for spare gates pattern. The default is ‘spare_*/*’.

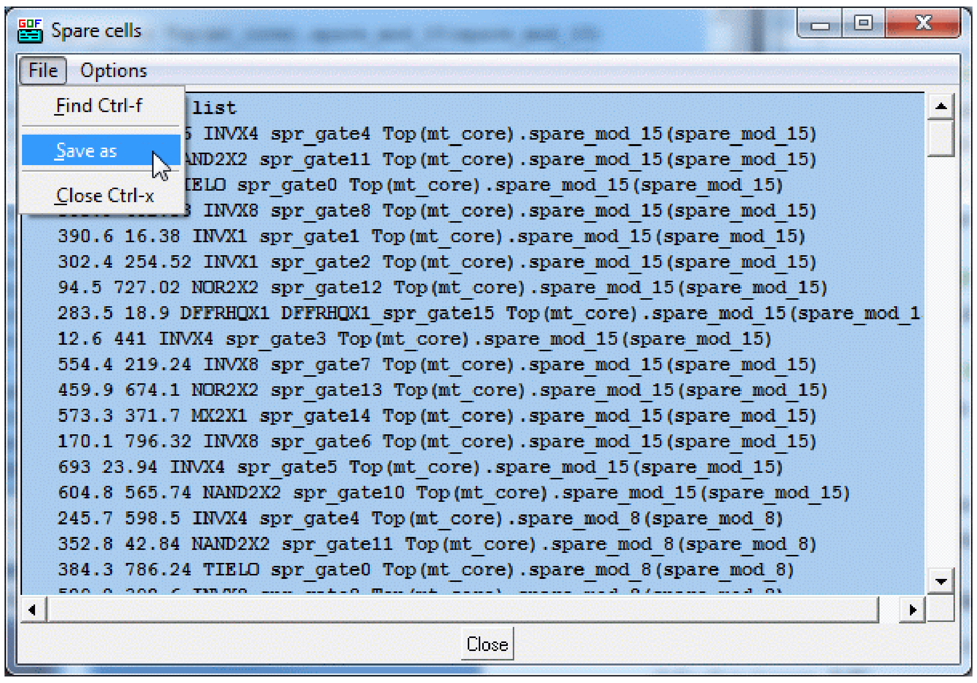


Spare Cell Pattern

Spare Cell Pattern spare_*/*

Figure 20 Spare Cell Pattern

Click 'OK' to extract spare instances from the netlist, and a pop text window appears to list all spare gate instances. Save the list to a spare list file for later usage.



```
 list
5 INVX4 spr_gate4 Top(mt_core).spare_mod_15(spare_mod_15)
AND2X2 spr_gate11 Top(mt_core).spare_mod_15(spare_mod_15)
IELO spr_gate0 Top(mt_core).spare_mod_15(spare_mod_15)
3 INVX8 spr_gate8 Top(mt_core).spare_mod_15(spare_mod_15)
390.6 16.38 INVX1 spr_gate1 Top(mt_core).spare_mod_15(spare_mod_15)
302.4 254.52 INVX1 spr_gate2 Top(mt_core).spare_mod_15(spare_mod_15)
94.5 727.02 NOR2X2 spr_gate12 Top(mt_core).spare_mod_15(spare_mod_15)
283.5 18.9 DFFRHQX1 DFFRHQX1_spr_gate15 Top(mt_core).spare_mod_15(spare_mod_1
12.6 441 INVX4 spr_gate3 Top(mt_core).spare_mod_15(spare_mod_15)
554.4 219.24 INVX8 spr_gate7 Top(mt_core).spare_mod_15(spare_mod_15)
459.9 674.1 NOR2X2 spr_gate13 Top(mt_core).spare_mod_15(spare_mod_15)
573.3 371.7 MX2X1 spr_gate14 Top(mt_core).spare_mod_15(spare_mod_15)
170.1 796.32 INVX8 spr_gate6 Top(mt_core).spare_mod_15(spare_mod_15)
693 23.94 INVX4 spr_gate5 Top(mt_core).spare_mod_15(spare_mod_15)
604.8 565.74 NAND2X2 spr_gate10 Top(mt_core).spare_mod_15(spare_mod_15)
245.7 598.5 INVX4 spr_gate4 Top(mt_core).spare_mod_8(spare_mod_8)
352.8 42.84 NAND2X2 spr_gate11 Top(mt_core).spare_mod_8(spare_mod_8)
384.3 786.24 TIELO spr_gate0 Top(mt_core).spare_mod_8(spare_mod_8)
```

Figure 21 Spare cell list

- Load Spare Cells File

  This command loads in the spare cells file created by the above command.

5.1.4.7 Launch Files Compare Window

This command launches netlist files compare window. The netlist compare function can do more than simple Linux 'diff' command.

5.1.4.8 Process Timing Violations

This command processes Prime Time report file. Read this PDF file for the use case.

www.nandigits.com/use_cases/gof_timing_analysis.pdf

5.1.4.9 SDF

This command group processes SDF file for delay information interactivity.

- Create SDF Index File

  GOF reads in SDF Index File instead of SDF file itself, since SDF can have huge size. Once the index file is created it can be reused next time and loaded in much faster.

- Load SDF Index File

  This command loads in already created SDF Index File.

5.1.5 Options Menu

5.1.5.1 Hierarchy Window Font

CheckGofViewer Figurefor the hierarchy list window position

- Increase Font Size

  Increase font size in the hierarchy list window.

- Decrease Font Size

  Decrease font size in the hierarchy list window.

5.1.5.2 Netlist Window Font

Netlist window locates in the right side of GofViewer window. CheckGofViewer Figurefor the netlist text window position

- Increase Font Size

  Increase the font size in netlist window.

- Decrease Font Size

  Decrease font size in netlist window.

5.1.5.3 Dump Waveform Restore File

An option window pops up for users to choose which dump restore file to be saved. It's useful for netlist simulation debug. When one format box is checked in pop menu, 'Write Waveform Restore File' item is presented on the top when one net is selected in the netlist window.

5.1.5.4 Setup

Integration of various setup information.

5.1.6 Help Menu

5.1.6.1 General

General help information.

5.1.6.2 About

About Gates On the Fly.

5.1.6.3 nandigits.com/gof_manual.htm

Visit the website for this manual.

5.1.6.4 Read Ethernet Mac Address

Read out MAC address. When users decide to purchase licenses or ask for evaluation licenses, MAC address is required to generate GOF licenses.

5.1.7 Keyboard Shortcuts

5.1.7.1 Access Menu

Press key 'Alt' and underlined letter in menu.

5.1.7.2 Functions access

- Ctrl-a: Select all text lines
- Ctrl-c: Copy the marked (highlighted) string
- Ctrl-v: Paste the content in clipboard
- Ctrl-d: Trace driver of the marked net
- Ctrl-f : Search function
- Ctrl-g: Load instance to schematic
- Ctrl-s: Emacs style search forward
- Ctrl-r: Emacs style search backward
- Ctrl-w: Write to waveform dump restore file
- Ctrl-x: Exit the current window

5.1.8 Selection Status

Click mouse-left-button on netlist text window, the object which can be net, instance or module under the cursor is highlighted. Netlist window pop menu has different content according to the selection status. Pressing keys Ctrl-a can have all content in the netlist window selected. Press mouse-left-button and don't release, move mouse down to select multiple lines.

5.1.9 Netlist Window Pop Menu

Click mouse-right-button and release, a pop menu appears under the cursor. The menu content varies with the selection status in the netlist window.

5.1.9.1 Search

Search for a string in the netlist window. Keyboard shortcut is Ctrl-f.

5.1.9.2 Copy Selected to

Copy the selected object (net or instance) to new schematic window or existing schematic window.

- Schematic New

  Copy the selected object to a new schematic.

- Schematic #number

  Copy the selected object to an existing schematic window.

5.1.9.3 Driver of the selected net

Trace to the driver of the selected net. The netlist window will show the instance that drives the net and mark the driven net.

5.1.9.4 List Connectivity of the selected net

Pop up a window to list the connectivity of the selected net.

5.1.9.5 List Fanin EndPoints

Pop up a window to list the fanin endpoints including flops and input ports that drive the selected net.

5.1.9.6 List Fanout EndPoints

Pop up a window to list the fanout endpoints including flops and output ports that are driven by the selected net.

5.1.9.7 Parent Module

Go to the definition location of the parent module calling the current module. It's only active in sub-modules, not in root top level module.

5.1.9.8 List Context

List the context of the selected object which can be net, instance or module.

5.1.10 Hierarchy Window Pop Menu

Click mouse-right-button and release, a pop menu appears under the cursor. The menu content varies with the selection status in the hierarchy window.

5.1.10.1 Show Definition

Open the module content and display it in the netlist window.

5.1.10.2 Show Calling

Open the parent module and highlight the instantiation location.

5.1.10.3 Report Area of the selected design

SeeReport Area

5.1.10.4 Report Leakage of the selected design

SeeReport Leakage

5.1.10.5 Report Leaf Cells of the selected design

SeeReport Leaf Cells

5.1.10.6 Report Submodules of the selected design

SeeReport Submodules

5.1.10.7 Statistic of the selected design

SeeStatistic of Current Design

5.1.10.8 Edit Module of the selected design

Edit the module by using edit tool defined in menu Options->Setup->Misc->'Edit tool'. It will ask for a directory for storing temporary files.

5.1.10.9 Save Module of the selected design

After editing, the edited modules can be saved into a file.

5.1.10.10 Goto Line Number#

5.2 GofTrace

GofTrace is an incremental schematic engine. Users control how to expand the schematic by clicking the input/output pins of gates with mouse-middle-button. Users can adjust the positions of the gates on the schematic any time by mouse-left-button.

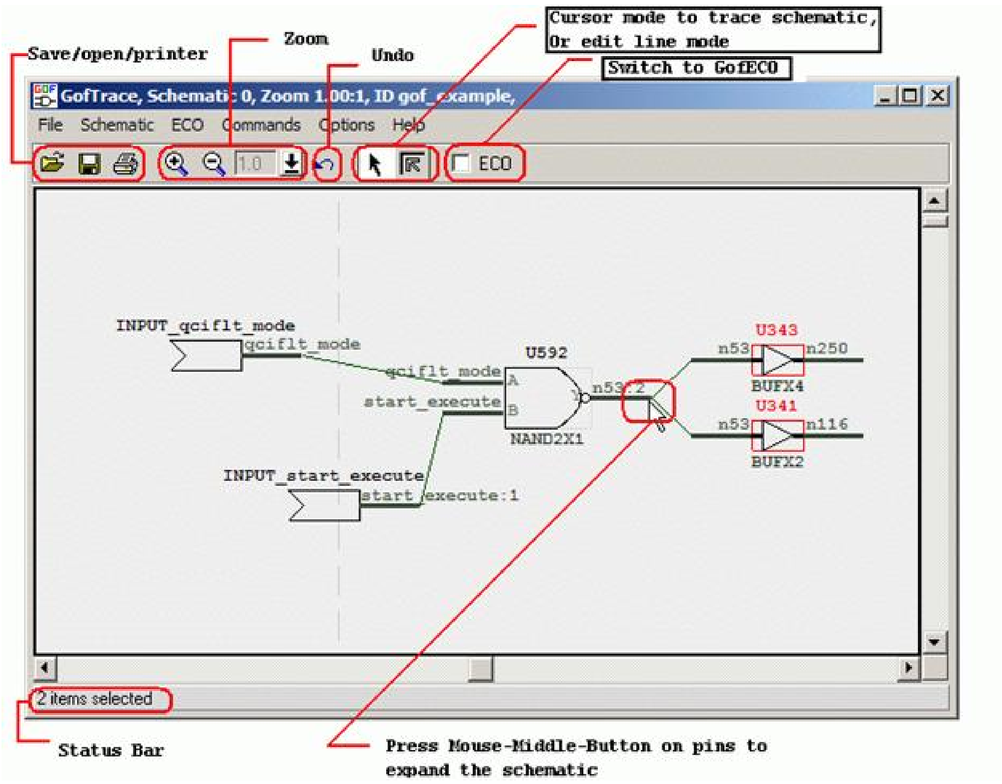


Figure 22 GofTrace Window

5.2.1 Mouse buttons usage

5.2.1.1 Mouse Left Button

Mouse left button is used to select object. Click on any object, it is highlighted to indicated being selected. Press 'ctrl' key and click on objects to select multiple objects. Press mouse left button and move the mouse to select multiple objects at one time.

5.2.1.2 Mouse Middle Button

Mouse middle button is used to trace the schematic. Click on input/output pins to expand the schematic. It used to do drag-drop function as well. In ECO mode, it's used to connect floating input pin to existing nets.

5.2.1.3 Mouse Right Button

Mouse right button is to popup menu.

5.2.2 File Menu

5.2.2.1 Save

Save the schematic to a file for future usage. The saved file has extension '.st' which can only be used by GOF in 'Open' schematic command shown below.

5.2.2.2 Open

Open schematic stored by Save command above.

5.2.2.3 Print

Print schematic to a printer or file. Printer Page Setup window pops up for the print scope setup. In Windows platform, users can select one of the printers configured in the system. In Linux platform, make sure 'lpr' command works.

5.2.2.4 Exit

Exit GofTrace window.

5.2.3 Schematic Menu

5.2.3.1 New Schematic

This command launches a new GofTrace schematic window.

5.2.3.2 List Gate

This command pops up a window for user to enter a string into the entry to find the matching instances. It accepts wildcards in both hierarchy name and instance name. For example, there are four hierarchical instances u_lane0, u_lane1, u_lane_2, u_lane3, each instance has spare modules with instance naming 'u_spare*', and in each spare module AND gate has instance naming '*AND*'. In order to find all spare AND gates, one can enter a string 'u_lane*/u_spare*/*AND*'.

5.2.3.3 Load Gate

This command pops up a window for user to enter a string into the entry to load the matching instances onto the schematic. Same as 'List Gate' command above, it accepts wildcards. However, the total number of gates drown on the schematic should not exceed the threshold defined in Menu Options->Setup->Misc->'Gates number limit'.

5.2.3.4 Load Gate Driving Net

This command pops up a window for user to enter a string into the entry as the net name. The tool will find the driver of the net and draw the driver on the schematic.

5.2.3.5 List Selected Instances

Use mouse-left-button to select a bunch of objects (Instances or wires) on the schematic. Click this command to list all the selected instances' full hierarchical names in a pop window.

5.2.3.6 List Selected Wires

Use mouse-left-button to select a bunch of objects (Instances or wires) on the schematic. Click this command to list all the selected wires' full hierarchical names in a pop window.

5.2.3.7 List Selected Modules

Use mouse-left-button to select a bunch of objects (Instances or wires) on the schematic. Click this command to list all the selected gates' module name in a pop window.

5.2.3.8 List Selected Instances Definitions

Use mouse-left-button to select a bunch of objects (Instances or wires) on the schematic. Click this command to list all the selected instances' full definitions in a pop window.

5.2.3.9 List Selected Gates Types

Use mouse-left-button to select a bunch of objects (Instances or wires) on the schematic. Click this command to list logic type numbers of all the selected gates in pop window. For example, 'AND' gate has type 'and', inverter has type 'not'. The pop window can have information such as "Type 'not' has 11".

5.2.3.10 Zoom In

This command can zoom in the schematic view. The maximum zoom in ratio is 100%. Keyboard shortcut for this command is key '+'.

5.2.3.11 Zoom Out

This command can zoom out the schematic view. The minimum zoom out ratio is 13%. Key board shortcut for this command is key '-'.

5.2.3.12 Zoom to

This command can directly select zoom ratio, the valid values are 100%, 67%, 44%, 30%, 20% and 13%.

5.2.3.13 Find Gates on Schematic

This command pops up a window for users to enter a string to find the matching instances on the schematic. It matches portion of the full name. For example, 'U' matches 'U0', 'U1' and 'U222'.

5.2.3.14 Find Nets on Schematic

This command pops up a window for users to enter a string to find the matching wires on the schematic. It matches portion of the full name. For example, 'Net0' matches 'Net0', 'Net011' and 'Net023'.

5.2.3.15 Undo Schematic Operations

This command is to undo schematic operations. Keyboard shortcut is Ctrl-z.

5.2.3.16 Place & Route

This command group is to automatically place the gates on the schematic and automatically route the wires.

- Auto Place & Route

  This command is to do both placement and routing automatically.

- Auto Place

  This command is to do automatic placement only.

- Auto Route

  This command is to do automatic routing only.

- Reset Route

  This command is to reset all existing routes, all routed wires become straight.

5.2.3.17 Create PS/PDF File

This command is to create Postscript file or PDF file for the current view of the schematic. In Windows platform, only Postscript is support.

On Linux platform both Postscript and PDF are supported.

5.2.4 Commands Menu

5.2.4.1 View Gates in Layout

This command launches layout viewer window. If some gates and wires are selected on the schematic, they will be highlighted on the layout viewer. It requires DEF and LEF physical design files to be loaded.

5.2.4.2 Load Layout Files

This command is to load layout physical design files. They include DEF and LEF files. The command can be run several times to load the physical design file one by one. DEF and LEF files can be loaded by command line with -def and -lef options. Or they can be read in by API 'read_def' and 'read_lef' in GofCall script.

5.2.4.3 Launch GofCall Script Interface

This command launches GofCall Script Interface window.

5.2.4.4 Spare Cells

This command group handles spare cells in automatic metal ECO flow.

- Create Spare Cells File

  This command creates spare cells file.

- Load Spare Cells File

  This command loads the spare cells file created by the command above.

5.2.4.5 Launch Files Compare Window

This command launches netlist files compare window.

5.2.4.6 Process Timing Violations

This command launches Prime Time report file processing controller.

5.2.4.7 SDF

This command group handles SDF loading.

- Create SDF Index File

  This command creates SDF file index file. GOF doesn't load the full SDF file, but SDF index file instead.

- Load SDF Index File

  This command loads SDF index file created by the command above.

5.2.5 Options Menu

5.2.5.1 Increase Font Size

This command increases the font size on the schematic.

5.2.5.2 Decrease Font Size

This command decreases the font size on the schematic.

5.2.5.3 Show Port

This option makes port name visible.

5.2.5.4 Show Wire

This option makes wire name visible.

5.2.5.5 Show Title

This option makes gate title visible.

5.2.5.6 Show Type

This option makes gate type visible.

5.2.5.7 Show Connections

This option makes wires visible.

5.2.5.8 Show Comment

This option makes comments visible.

5.2.5.9 Dump Waveform Restore File

This command pops up a window to setup simulation waveform restore file. Four waveform restore file formats are supported.

- SimVision Restore File
- ModelSim Restore File
- Verdi Restore File
- GtkWave Restore file

If one or more of the formats are selected, GofViewer and GofTrace pop menus will have 'Write Selected Nets to the Waveform Restore File' as the first item, when a net is selected.

5.2.5.10 Save String to Clipboard

This option enables saving string to clipboard when a wire or instance name is clicked by mouse-left-button.

5.2.5.11 Cursor Mode

This is normal mode of the schematic tracing.

5.2.5.12 Line Edit Mode

This mode sets cursor in editing wire connections mode. Press mouse-left-button on the straight wire connection and move, the line will be pulled by the cursor until the mouse button is released.

5.2.5.13 Setup

The command pops up configuration window for the tool setup.

5.2.6 Help Menu

5.2.6.1 General

General help information.

5.2.6.2 About

About Gates On the Fly.

5.2.6.3 nandigits.com/gof_manual.htm

Visit the website for the manual.

5.2.7 Keyboard Shortcuts

5.2.7.1 Access Menu

Press key 'Alt' and underlined letter in menu.

5.2.7.2 Functions access

- Ctrl-a: Select every object on the schematic
- Ctrl-c: Copy the selected objects
- Ctrl-v: Paste the content in clipboard copied by Ctrl-c
- Ctrl-g: Load instance to schematic
- Ctrl-w: Write to waveform dump restore file
- Ctrl-x: Exit the current window
- Ctrl-digit: Save the selection location as the digit indication, the schematic view will move the current saved location when the digit is pressed later

5.2.8 Selection Status

Click mouse-left-button on the schematic window, the object which can be net, instance under the cursor is highlighted. GofTrace pop menu has different content according to the selection status. Pressing keys Ctrl-a can have all selected on the schematic. Press mouse-left-button on empty space, and don’t release, move mouse down to select multiple objects.

5.2.9 GofTrace Pop Menu

Click mouse-right-button on GofTrace schematic, a menu pops up. The content of the menu varies as the selection status on the schematic.

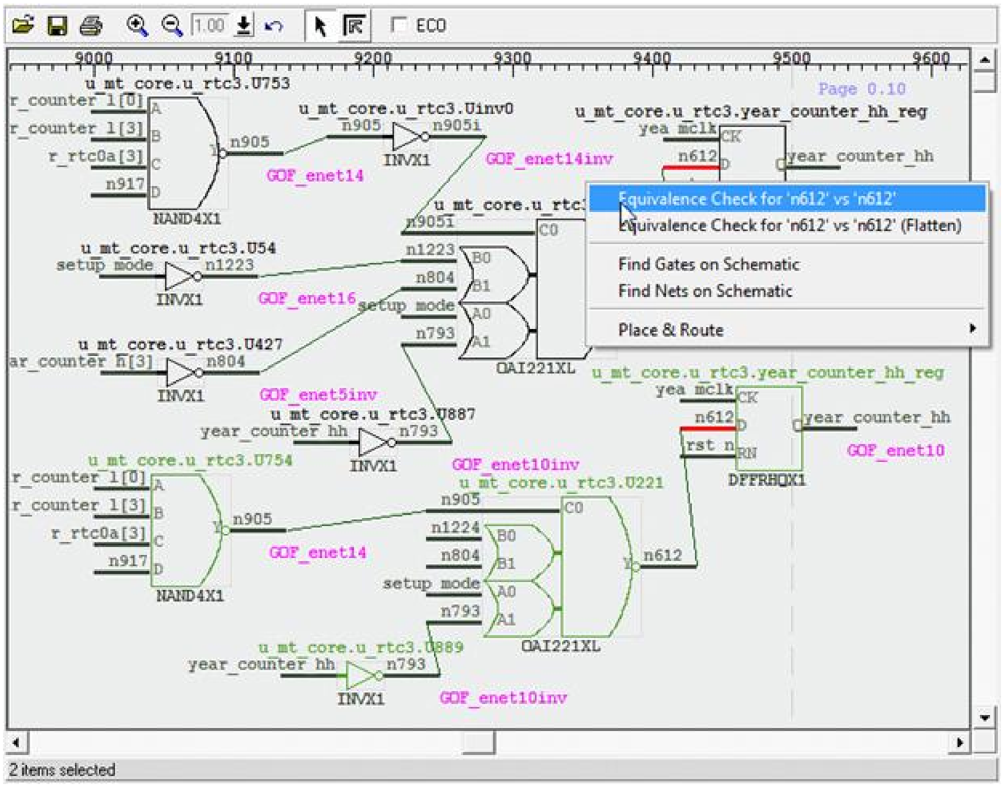
5.2.9.1 Nets Equivalence Check

The command needs Reference Netlist loaded.

- Reference Netlist can be loaded by '-ref' option in command line. For example, "gof -lib tsmc.lib implementation.v -ref reference.v"
- Reference Netlist can be loaded by 'read_design' with '-ref' switch in GofCall script. For example, 'read_design("-ref", "reference.v")'

Use mouse-left-button to select on a pin in implementation netlist and press 'ctrl' key to click mouse-left-button on the other pin in reference netlist. So that one pin in implementation netlist and the other comparing pin in reference netlist are selected at the same time.

Click mouse-right-button to popup menu and select "Equivalence Check for 'neta' vs 'netb'" command.



2 items selected

Figure 23 Nets Equivalence Check

When the check is done, a pop window shows if the nets are equivalent. After the check is done, the tool keeps all equivalence information for every instance in the logic cone. Use mouse-middle-button to expand the schematic, the information is marked as 'GOF_enet###'. It's very convenient for Logic Equivalence Failure debug. For example, in Figure 23, 'u_mt_core.u_rtc3.Uinv0' was added by mistake in a Timing ECO and it was caught by the tool by showing as inverted logic vs the reference design.

5.2.9.2 Find Logic Equivalent Nets

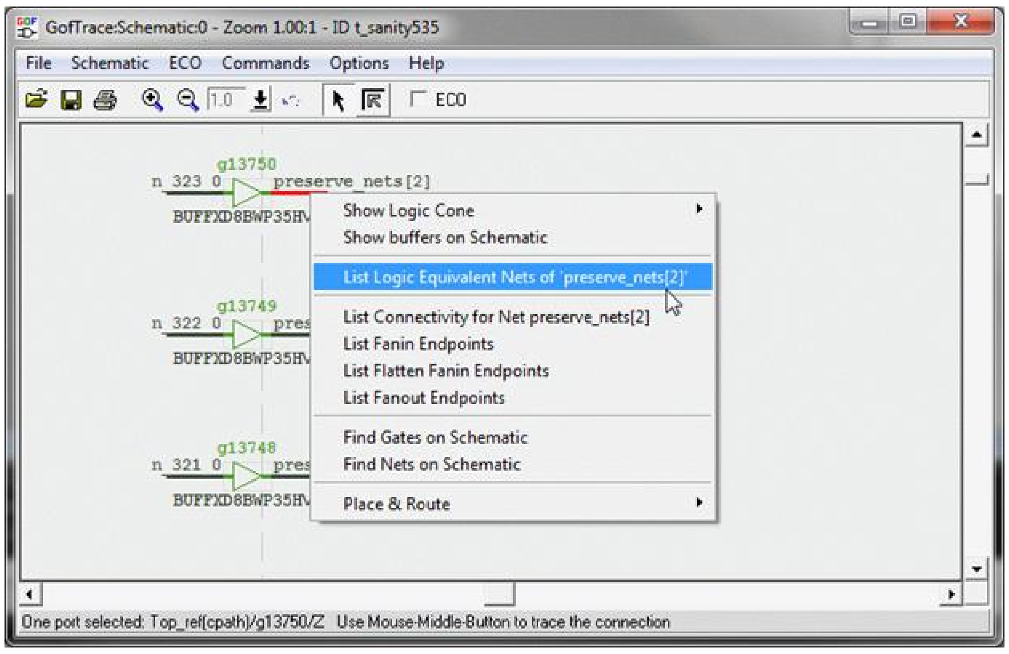
Wire names in RTL codes are normally optimized away in synthesis process. GOF has a way to retrieve them. A reference netlist with the wire names preserved should be synthesized. There are several ways to preserve wire names.

- Put the wires in output ports
- Use 'preserve' or 'dont_touch' command in synthesis constraint file

Reference Netlist should be loaded with Implementation Netlist.

- Reference Netlist can be loaded by '-ref' option in command line. For example, "gof -lib tsmc.lib implementation.v -ref reference.v"
- Reference Netlist can be loaded by 'read_design' with '-ref' switch in GofCall script. For example, 'read_design("-ref", "reference.v")'

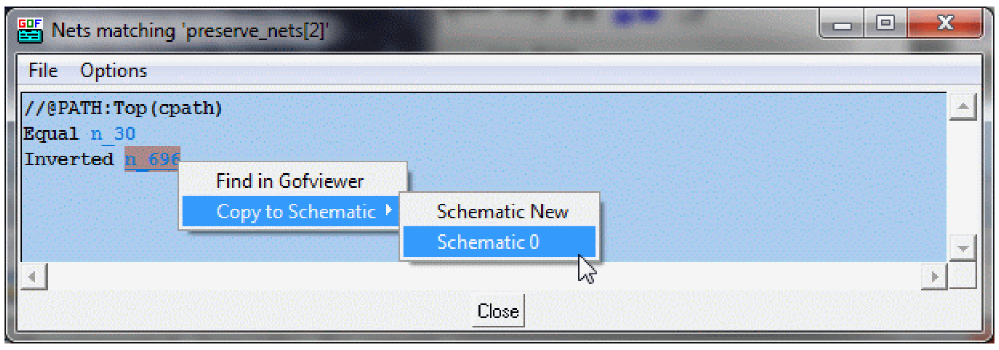
Start GUI Window, load the preserved wires on a schematic window. Select the pin that drives the wire, right click mouse-right-button to popup menu. Select 'List Logic Equivalent Nets of ...' command.



One port selected: Top_ref(cpath)/g13750/Z Use Mouse-Middle-Button to trace the connection

Figure24 List Logic Equivalent Nets

A popup window appears to list the nets that Logically Equal or Invert to the reference net. By popup menu on the window, the corresponding nets can be easily loaded into the schematic.



```
//@PATH:Top(cpath)
Equal n_30
Inverted n_696
```

Figure25 Logic Matching Nets

The feature is very convenient in manual ECO when the optimized wires have to be located in Implementation Netlist.

5.2.9.3 Find Circuit between Two Points

When two or more than two instances are selected on the schematic, the command item appears in the pop menu. This command finds the combination logic path between two instances.

5.2.9.4 Add Comments

This command adds comments entered by users on the schematic.

5.2.9.5 Find Gates on Schematic

See Find Gates on Schematic.

5.2.9.6 Find Nets on Schematic

See Find Nets on Schematic.

5.2.9.7 Place & Route

See Place & Route.

5.2.9.8 Show Logic Cone

Logic Cone is the logic cluster between flops and ports, as shown in the following figure. Users should select the output flop or its pins to do logic cone extraction.

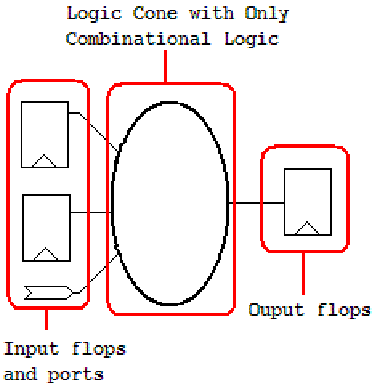


Figure26 Logic Cone

- On the Schematic

  Draw the whole logic cone on the schematic.

- In Text Mode

  Display the whole logic cone in a pop up text window.

5.2.9.9 Copy Selected to

This command group does interactions between GofTrace windows and LayoutViewer windows.

- Schematic New

  Copy the selected items to a new schematic.

- Schematic Number#

  Copy the selected items to an existing schematic identified by ID Number.

- Layout New

  Copy the selected items to a new launched LayoutViewer window. The selected circuit will be marked on the LayoutViewer window.

- Layout Number#

  Copy the selected items to an existing LayoutViewer window indentified by ID Number.

5.2.9.10 Find selected in GofViewer

This command finds the selected instance back in GofViewer netlist window, and highlights the instance in the netlist window.

5.2.9.11 Edit Gate Display

This command pops up a window for users to add or change comments associated with the gate and change the color of the gate.

5.2.9.12 List Logic for the Selected Leaf Cell

This command pops up a text window to list the logic of the selected leaf cell.

5.2.9.13 List Context for the Selected Leaf Cell

This command pops up a text window to list the library content of the selected leaf cell. The content includes the cell's pin definitions, area and timing.

5.2.9.14 List Definition for the Selected Instance

This command pops up a text window to list the instantiation of the selected instance.

5.2.9.15 Load Instance Similar to the Selected Instance

This command pops up an entry window with the current selected instance name pasted in the entry. So that user can do simple change to load other similar naming style instance onto the schematic.

5.2.9.16 Equivalent Symbol

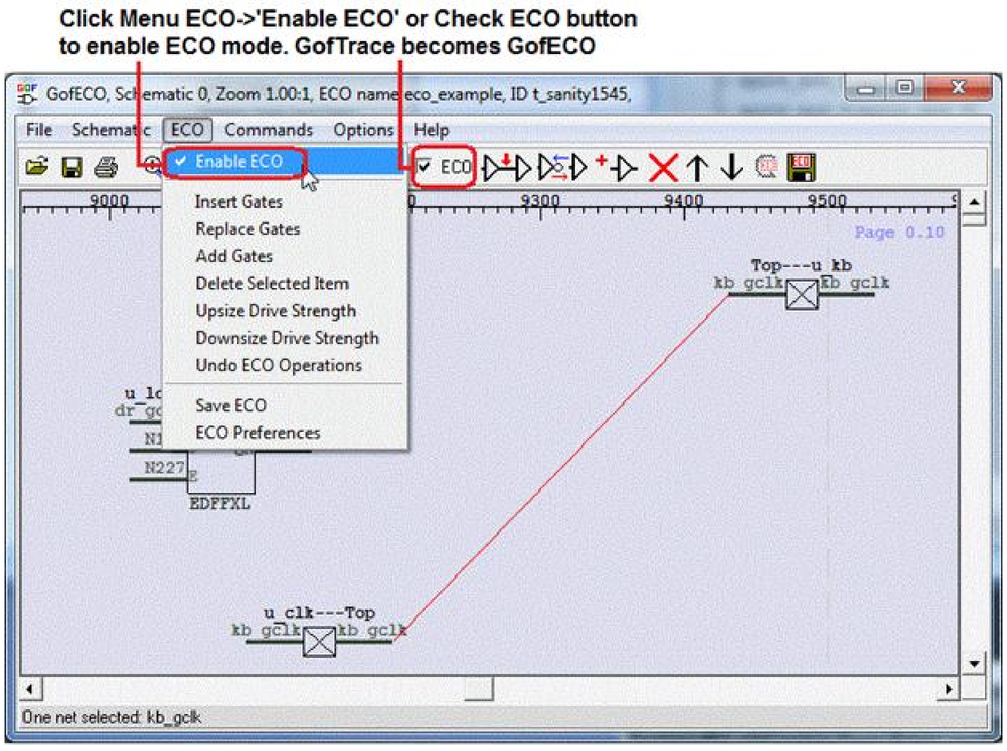
This command changes the selected gate symbol display to the**equivalent symbol according to DeMorgan's Laws. For example, NAND symbol is equivalent to Inputs Inverted OR symbol.**

5.2.9.17 Delete

This command deletes the selected objects on the schematic. The object can be gates, wires and comments.

5.3 GofECO

GofECO uses the same window as GofTrace by enable ECO mode. The background color changes to light blue by default. The color can be configured by Menu Setup->GofECO->Color->BackGround. The ECO operation icons appear on the tool bar. GofECO uses the same menus GofTrace uses, besides the contents in ECO menu being activated.



Click Menu ECO->'Enable ECO' or Check ECO button to enable ECO mode. GofTrace becomes GofECO

GofECO, Schematic 0, Zoom 1.00:1, ECO name eco_example, ID t_sanity1545,

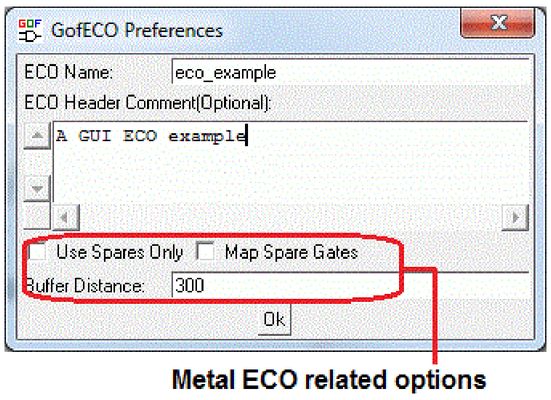
One net selected: kb_gclk

Figure 27 GofECO Window

5.3.1 ECO Menu

5.3.1.1 Enable ECO and ECO Preferences

This option enable ECO mode, GofTrace switches to GofECO. A pop up window appears for inputting ECO setups.



GofECO Preferences

ECO Name: eco_example

ECO Header Comment(Optional):

A GUI ECO example

□ Use Spares Only □ Map Spare Gates

Buffer Distance: 300

Ok

Metal ECO related options

Figure 28 ECO Preferences

- ·ECO Name should be unique, so that name confliction can be avoided
- ·ECO Header Comment is optional, which will appear at the beginning of ECO netlist file
- ·Checkbuttons 'Use Spares Only' and 'Map Spare gates' and 'Buffer Distance' entry are for Metal Only ECO. Their usages are:
- Use Spares Only' is to use spare type gates only, a spare gate list file must be loaded with this option enabled.
- Map Spare Gates' is to let the tool mapping any type gates to either the spare type gates or the exact spare instances in the design.
- Buffer Distance' entry is to tell the tool add buffers/repeaters when the connection distance is larger than the limit. Inputting a large number can disable adding buffers. The corresponding script command is 'set_buffer_distance'.

5.3.1.2 Insert Gates

This command inserts gates in the selected wires. It requires one or more wires being selected on the schematic, before inserting gates. A 'Gate selection' window pops up for users to select proper type of gates and gate number. When multiple wires are selected and some wires have the same drivers, users can choose either one gate driving all shared wires or one gate driving each wire. Users will be asked to choose the pin connections in 'Specify pin connections' window. The default pin connections setup can be used and users can modify the connections later on the schematic. Read this PDF use case for more detail.

http://www.nandigits.com/use_cases/gof_insert_buffers_inverters.pdf

5.3.1.3 Replace Gates

This command replaces the selected gates with a different type of gates. It requires one or more gates being selected on the schematic. If two or more than two gates are selected, they should have the same type. A 'Gate selection' window pops up for users to select proper type of gates to replace the selected ones. Users will be asked to choose the pins connections in 'Specify pin connections' window. The default pin connections setup can be used and users can modify the connections later on the schematic.

5.3.1.4 Add Gates

This command adds new ECO gates on the schematic. A 'Gate selection' window pops up for users to select proper type of gate to add onto the schematic. The new ECO gates appear as output driving a new net and input floating. The hierarchy of the gate is undefined. When users connect one of the input pins to another existing gate or connect other gate's floating input to the ECO gate's output pin, the ECO gate gets the same hierarchy as the other gate. ReadAdd Connectionfor more detail.

5.3.1.5 Delete Selected Items

This command deletes the selected items. Users would be warned for deleting multiply objects at the same time.

5.3.1.6 Upsize Drive Strength

This command upsizes the selected gate to a higher drive strength gate with the same type. If there is no higher drive strength gate available, users would be warned with a pop up information window.

5.3.1.7 Downsize Drive Strength

This command downsizes the select gate to a lower drive strength gate with the same type. If there is no lower drive strength gate available, users would be warned with a pop up information window.

5.3.1.8 Undo ECO Operations

This command undoes the previous ECO operation, until no more ECO operation is in the pipeline.

5.3.1.9 Add Connection

There is no operation button/icon for Add Connection operation. Adding connection can only be done from a floating input pin to a output pin. User can press mouse-middle-button on a floating input pin, and don't release the mouse. Then move mouse to the destination output pin of the instance that user would like the wire connected to, release the button to make the connection to be created.

5.3.1.10 Save ECO

This command saves ECO result to a file. The supported file formats:

- Verilog netlist
- GofCall Perl Script
- SOC Encounter ECO script
- Tcl script for Synopsys
- DC script for Synopsys

5.3.2 MetalOnlyECO

Metal ECO only touches metal layers. Gates On the Fly provides four Metal Only ECO modes by combinations of setting up the options inECO preferenceand loading DEF file.

5.3.2.1 Metal ECO, mode 1

User can add any type of gates and let the tool map to the spare type gates, Place and Route tool should map the spare type gates to the exact spare gate instances.

The setup for this mode:

- Spare gate list file should be created and loaded.
- DEF file should NOT be loaded.
- 'Use Spares Only' is NOT checked.
- 'Map Spare Gates' is checked.

A use case for Metal Only ECO mode 1 can be found in

http://www.nandigits.com/use_cases/gof_gui_metal_eco_mode_1.pdf

5.3.2.2 Metal ECO, mode 2

User can add any type of gates and let the tool map to the exact physically existing spare gate instances.

The setup for this mode:

- Spare gate list file should be created and loaded.
- DEF file should be loaded.
- 'Use Spares Only' is NOT checked.
- 'Map Spare Gates' is checked.

A use case for Metal Only ECO mode 2 can be found in

http://www.nandigits.com/use_cases/gof_gui_metal_eco_mode_2.pdf

5.3.2.3 Metal ECO, mode 3

User can add only spare type gates and let the tool map to the exact spare gate instances.

The setup for this mode:

- Spare gate list file should be created and loaded.
- DEF file should be loaded.
- 'Use Spares Only' is checked.
- 'Map Spare Gates' is checked.

A use case for Metal Only ECO mode 3 can be found in

http://www.nandigits.com/use_cases/gof_gui_metal_eco_mode_3.pdf

5.3.2.4 Metal ECO, mode 4

User can pick the exact spare gate instances, and connect and disconnect up the instances in ECO.

The setup for this mode:

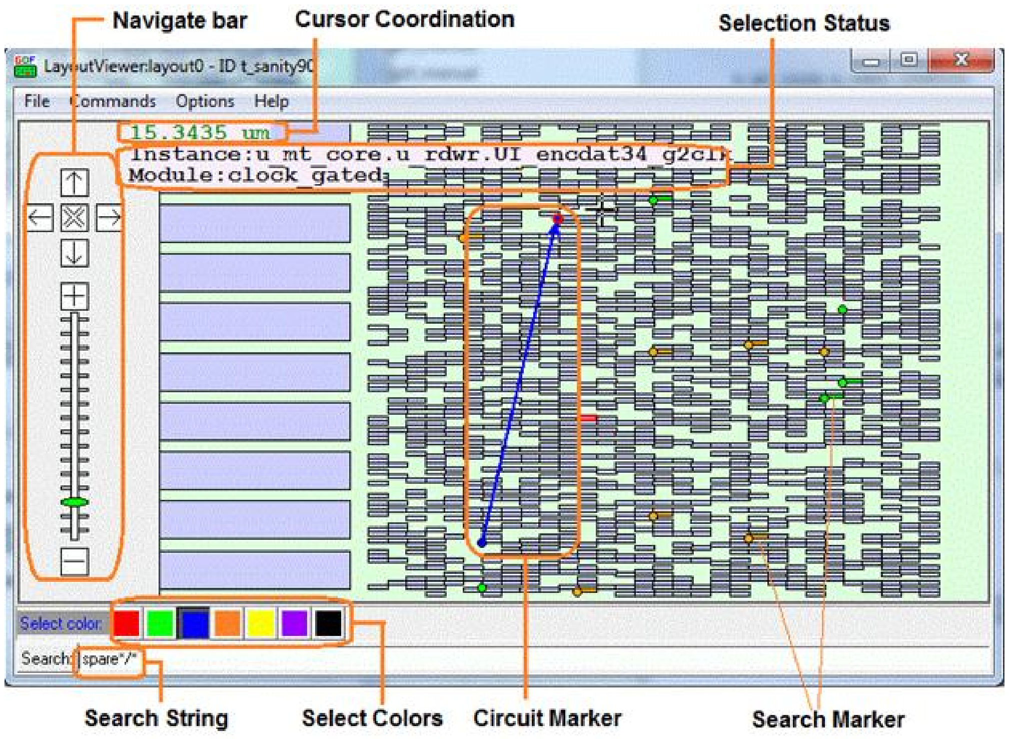
- Spare gate list file has no need to be created and loaded.
- DEF file should be loaded.
- 'Use Spares Only' is NOT checked.
- 'Map Spare Gates' is NOT checked.

A use case for Metal Only ECO mode 4 can be found in

http://www.nandigits.com/use_cases/gof_gui_metal_eco_mode_4.pdf

5.4 LayoutViewer

LayoutViewer window displays partial physical placements. The circuit drawn on the schematic can be highlighted on LayoutViewer. It has full interactivity with GofTrace. It requires physical design files including DEF and LEF files to be loaded.



15.3435 um

Instance:u_mt_core.u_rdwr.UI_encdat34_g2clk

Module:clock_gated

Search: spare*/*

Figure 29 LayoutViewer Window

5.4.1 File Menu

5.4.1.1 Capture in PDF

This command captures the current LayoutViewer display to PDF file. PDF is only supported in Linux Platform. In Windows Platform, the captured display is saved in PostScript format.

5.4.1.2 Exit

Exit LayoutViewer.

5.4.2 Commands Menu

5.4.2.1 Clear Circuit Markers

Clear circuit markers which can be created by Drag-And-Drop from GofTrace Partial Schematic.

5.4.2.2 Clear Search Markers

Clear search markers which are those highlighted cells matching the searching string in search entry.

5.4.2.3 New Schematic

When cells or markers are selected in LayoutViewer, this command can launch a schematic with selected instances on it.

5.4.3 OptionsMenu

5.4.3.1 Show Grid

This option shows grid on LayoutViewer.

5.4.3.2 Show Instance

This option shows instance name on LayoutViewer. Zoom in scale should be large enough to show instance names.

5.4.3.3 Show Module

This option shows module name on LayoutViewer. Zoom in scale should be large enough to show module names.

5.4.3.4 Setup

LayoutViewer setups which include maximum search matching number and placement display zone area size.

5.4.4 Help Menu

5.4.4.1 Help on LayoutViewer

Visit NanDigits web site for Gates On the Fly manual section LayoutViewer.

5.4.5 LayoutViewer Pop Menu

Click mouse-right-button to pop up the menu.

5.4.5.1 Clear Circuit Markers

Clear circuit markers which can be created by Drag-And-Drop from GofTrace Partial Schematic .

5.4.5.2 Clear Searching Markers

This command clears searching markers which were activated by search function.

5.4.5.3 Copy Selected to

This command copies the selected gates to the following destination:

- Back to GofViewer
- A new Schematic
- An existing Schematic indentified by Number ID

5.4.6 Keyboard and mouse combination

5.4.6.1 Ctrl key to measure length

Press 'Ctrl' key and move mouse, the Cursor Coordination displays the length cursor moves in unit of 'um'.

5.4.6.2 Shift key to select multiple markers

Press 'Shift' key and press mouse-left-button, move mouse to draw a virtual rectangle. When release the mouse-left-button, those markers in the virtual rectangle are all highlighted. Click mouse-right-button to pop menu, those selected instances can be sent to other schematics or GofViewer the netlist view window by 'Copy Selected to' command.

5.4.7 Mouse operations

- Mouse-middle-wheel: Roll up to zoom in and roll down to zoom out the LayoutViewer window.
- Mouse-left-button: Click and release to select cells or markers. Press on LayoutViewer window to move it around.
- Mouse-middle-button: Drag-And-Drop selected instances.
- Mouse-right-button: Release to pop up menu.

5.4.8 Select color buttons

Click color buttons in 'Select color:' bar to select the current color. 'Select Color:' string changes to the current selected color. Any new Circuit Markers and Search Markers will have this color.

5.4.9 Search function

Type search string in Search Entry to highlight the leaf instances matching the string on the LayoutViewer. The search string is in 'path/instance' string format, separated by '/'. Wildcard can be used in path and instance names. The markers have the color selected in 'Select color' bar.

The search string takes these options:

- -spare: When spare gate list is loaded by -sparelist option or get_spare_cells command.
- -type type-name: Only search those instances with specified type, 'nand' for example.
- -hier: Search all leaf instances under the specified hierarchy. For example, 'u_clk/* -hier'.
- -ref ref-name: Only search those instances with specified reference, 'NAND2X2' for example.

Examples:

'u_rtc/*' : Search leaf instances in hierarchy 'u_rtc'.

'* -hier -type nand': Search all leaf instances with 'nand' type in the design.

'u_clk/* -hier': Search all leaf in hierarchy 'u_clk' and its sub-hierarchies.

6 Appendix A

6.1 GOF Command Options

```
gof [options] netlists
options:
-h: Print out this info.
netlists: Load netlist. There can be multiple netlist files listed,
if the design has more than one netlist files.
-lib technology_library: Or '-synlib technoloty_library', Load in technology library (synthesis library)
which has parameters for leaf gates.
There can be multiple -lib options,
if the design has more than one technology library files.
-v simulation_library: Specify simulation library file name which has Verilog definition
for leaf gates, like AND2X4.
There can be multiple -v options, if the design has more than one simulation library.
Both -v and -lib options are for leaf gates definition.
But -lib can support more features, like logic cone extraction.
In some cases, some leaf cells don't have technology library,
simulation library is used instead to resolve the cells.
-v option can coexist with -lib option.
-vmacro macro_library_file: For ECO purpose. Each module in the file appears as leaf cell, and it can be
added like other leaf cell in ECO.
When write out ECO netlist, the file content appear in the beginning of ECO netlist. And the
ECO cell is added as a hierarchical sub-block.
-y library_directory: Specify library directory, one file corresponds to one leaf gate.
+libext+.v should be used with -y option.
+libext+.v: Specify library file extension in library directory.
it can be +libext+.v+.vg+.vlib,
if .vg and .vlib are also lib extensions.
-run script_file.pl: Invoke GofCall to run netlist processing script, GOF will stay in shell mode when the
script finishes.
-vn library_name: Specify none leaf gates library file. For example, a file has
collection of macro cell which is defined by leaf gates.
-yn library_directory: The files in the directory are not defining leaf gates.
-shell: Run in text mode with shell prompt, GofCall APIs can be run in interactive mode in shell
-o log_file: Save log file, default gatesof.log
-Top_1 netlist_files: Load another netlist files to build Top_1 tree. The hierarchy will shown up in left
side of GofViewer window.
You can use -Top_2 -Top_3 ... to load more netlist files.
Note, when this option takes all netlist files followed, so the main netlist files should appear
before this option
For example, 'gof -lib tsmc.lib revised_netlist1 revised_netlist2 -Top_1 golden_netlist1
golden_netlist2' will create two trees in the left side of GofViewer window.
Note: ‘gof -lib tsmc.lib -Top_1 golden_netlist1 golden_netlist2 revised_netlist1
revised_netlist2' will build only one tree, since Top_1 option eats up all of the netlist files, the main tree is gone.
+define+PARAMETER0+PARAMETER1: Define PARAMETER0 PARAMETER1.
-id mydesign: Specify design id will appear on tile. Optional.
-def def_file: Load DEF, design exchange format, file.
```

-lef lef_file: Load Library Exchange Format file
-pdef pdef_file: Load in pdef file.
There can be multiple -pdef options,
if the design has more than one pdef files.
DEF file is preferred if both DEF and PDEF files are available.
-gtech gtech_file: Load in gtech file for wire matching.
-sparelist spare_cells_list_file: Load in spare cells list file.
-f file_list_file: Load all the files and options in the file_list_file
-vcd vcd_file: Load in VCD file for schematic annotation
-textbutton: Text mode button instead of image mode button in ECO operations
-version: Print out current version
-licquery: Query license usage

6.2 Command line Examples

gof -lib tsmc.lib soc.v
--- You have one netlist file 'soc.v' and one technology library, 'tsmc.lib'
gof -lib tsmc_std.lib -lib tsmc_io.lib top.v part0.v part1.v
--- You have three netlists, top.v, part0.v and part1.v, two technology library files
standard library cell, tsmc_std.lib, IO cells, tsmc_io.lib
gof -lib tsmc_std.lib -lib tsmc_io.lib -v analog_models.v top.v part0.v part1.v
--- Some analog cells having no technology library,
using '-v' option instead to load in analog simulation library
gof -lib tsmc_std.lib -lib tsmc_io.lib -vn macros.v -v analog_models.v top.v part0.v part1.v
--- macros.v has defined macro cells like sync-cells
gof -v /home/lib/tsmc_cells.v /home/netlist/soc.v
--- You would rather to use simulation library instead of technology library
gof -v /home/lib/tsmc_cells.v -v /home/lib/macro_cells.v +libext+.v -y /home/lib /home/netlist/top.v /home/netlist/soc.v
--- Multiple simulation library files, and some leaf cells are defined in '/home/lib' directory,
which uses '-y' option to resolve them
gof -v /home/lib/tsmc_cells.v +libext+.v -y /home/lib -yn /home/vmodule /home/netlist/soc.v
--- '/home/vmodule' has some macro cells files
gof -v /home/lib/tsmc_cells.v /home/netlist/soc0.v /home/netlist/soc1.v -id the_soc_design
--- You would like to display 'the_soc_design' as the title
gof -lib tsmc.lib -def soc.def.gz -lef libcell.lef soc.v
--- Design Exchange Format file soc.def.gz. And library exchange format file. For layout view usage
gof -v /home/lib/tsmc_cells.v /home/netlist/soc.v -gtech soc_submod.gtech.gv
--- Load in gtech file for RTL wire to netlist mapping
gof -lib tsmc.lib soc.v -run scripts.pl
--- Process netlist with scripts.pl in text mode, scripts.pl is in perl syntax
gof -lib tsmc.lib revised_netlist.v -Top_1 golden_netlist.v
--- Load both revised netlist and golden netlist, it's good for netlist comparison

7 Appendix B

7.1 Fatal codes

F-000: License failed
F-001: Time out in adding ports in hierarchies
F-002: Empty ID for nets
F-003: Pin connections processing fatal error
F-004: Net id not defined
F-005: Net is not in EpHash
F-006: Instance has not been mapped position in AUTO ECO
F-007: Instance has no name mapping in AUTO ECO
F-008: No net found for ECO instance/pin
F-009: Unknown connection type of instance/pin in AUTO ECO
F-010: Net has no name mapping in AUTO ECO
F-011: Failed to initialize database
F-012: MCell get sub-chains error
F-013: No tree has been defined
F-014: No ID for leaf cell pin
F-015: Undefined subroutine in GofCall script
F-016: Global symbol requires explicit package name
F-017: Syntax Error
F-018: Illegal Division by zero
F-019: Bare word not allowed
F-020: Can't locate Perl module
F-021: File size too large for evaluation mode
F-022: Internal error in make miss

7.2 Error codes:

E-001: Reference netlist has not been loaded
E-002: DEF file has missing section
E-003: Command line needs an option for a switch
E-004: Liberty files have not been loaded
E-005: Library cell doesn't exist
E-006: Delete middle bit in a bus
E-007: Unknown command line option
E-008: Win32 doesn't support .gz file
E-009: DEF file doesn't have DIEAREA item
E-010: Files loading sequence
E-011: Instance or pin or port can't be found in module
E-012: Net doesn't exists in module
E-013: Tree name doesn't exist
E-014: Hierarchical module name doesn't exist
E-015: Miss argument
E-016: Module stack is empty, too many pop_top
E-017: 'instance/pin' has wrong format
E-018: Instance or module doesn't exist
E-019: Instance doesn't have pin
E-020: Item is a black box
E-021: Missing DEF file
E-022: No reference for instance
E-023: 'leaf/pin' doesn't exist
E-024: Power connection format is wrong
E-025: Spare cell pattern is not specified
E-026: Spare list file doesn't exist
E-027: 'get_spare_cells' run before 'map_spare_cells'
E-028: 'instance/pin' is floating
E-029: New instance conflicts with existing one
E-030: Specify leaf:num in more than one output leaf
E-031: Instance should be leaf in change_gate
E-032: Syntax error in pin mapping
E-033: The new gate type should be different from the old one in change_gate
E-034: Leaf cell doesn't exist in libraries
E-035: Net doesn't have a driver
E-036: Instance name has special character that the tool doesn't support
E-037: Wrong argument in ECO APIs
E-038: Net has multiple drivers
E-039: Not a port
E-040: New port conflicts with existing one
E-041: Single bit wire can't be expanded to a bus
E-042: New port direction conflicts with existing one
E-043: Commands loading sequence
E-044: Nets in one ECO command should be in the same hierarchy
E-045: Missing scan control pins
E-046: Reference netlist is not loaded
E-047: Fail to open file for write
E-048: Fail to open file for read
E-049: Unable to recognize file format
E-050: Command line option needs a value
E-051: Path doesn't exist
E-052: Leaf should have only one output pin
E-053: New net conflicts with existing one
E-054: Instance ECO result not consistent
E-056: Net has no driver
E-057: Net has invalid BDD

E-059: Not enough resource to run synthesis
E-060: Not valid patch file
E-061: No spare cell for one gate type
E-062: Output port is driven by input port
E-063: Reference register doesn't exist in implementation netlist
E-064: No inverter in the database
E-067: Should add instance into fix_logic argument
E-071: Port doesn't exist in hierarchical instance
E-072: Black box instance doesn't exist in implementation netlist in AUTO ECO
E-076: Spare cells list file has Wrong format
E-080: GOF_KEY_FILE variable has not been defined
E-081: Use '-run' to run Perl script
E-082: Gtech file doesn't exist
E-085: Syntax error in netlist
E-101: No hierarchical path is used
E-102: Interrupt GUI operation by user
E-103: 'read_def' should be run before 'get_spare_cells'
E-105: Load specific file without the right option
E-106: Source ID can't be deleted
E-109: Found combinational loop
E-110: Implementation Netlist has not been loaded
E-112: Can't find pin direction

7.3 Warning codes

W-001: Bypass already loaded file
W-002: DEF has some section missing
W-003: DEF has module not resolved
W-004: No ECO pin specified for ECO instance
W-005: Not enough spare cells
W-006: DEF file not loaded
W-007: Leaf cell doesn't have timing table
W-023: 'leaf/pin' doesn't exist
W-028: 'instance/pin' is floating
W-038: Net has multiple drivers
W-054: Instance ECO result not consistent
W-055: Net ECO result not consistent
W-056: Net is not driven
W-060: Invalid patch file
W-061: No spare cell for one gate type
W-065: Tie floating input pin to zero
W-066: New port created in AUTO ECO
W-068: Hierarchical cell is created in AUTO ECO
W-069: Set don't touch Warning
W-070: Can't find repeaters
W-073: 'instance/pin' is inverted but being forced to be equal by user
W-074: 'instance/pin' is forced to be inverted by user
W-075: Net returned wrong BDD
W-077: No size information for a leaf
W-078: Module is redefined
W-079: Instance can't be resolved in GTECH
W-080: Leaf cell can't be resolved in module
W-083: Can't read MAC Address
W-084: Sub-module can't be resolved
W-086: Include file doesn't exist
W-087: Bit-width mismatch in instantiation
W-088: Zero fanin endpoint
W-089: Can't find ECO instance position
W-090: Empty instance name in patch file
W-091: ECO net has no fanout
W-092: New input port created and needs to be connected
W-093: New ID created for end point
W-094: Can't detect port phase in module
W-095: Port or net is forced to be equal by user
W-096: Port and net has mismatching bit-width
W-097: Schematic only feature
W-098: Force to use 1'b0/1'b1 in AUTO ECO
W-099: Can't fix timing, since lacking valid points
W-100: No lib name for a leaf cell
W-104: Module is defined as leaf cell but has definition in the netlist
W-107: Module is set as a leaf by user
W-108: Module is not uniquified
W-111: No need to set path prefix
W-112: Can't find pin direction
W-113: Different types of flops in IMP and REF

7.4 GUI warning codes

GW-001: Don't connect net to a new created connector
GW-002: Don't connect two ECO connectors
GW-003: Don't drive an output port by a cell in different hierarchy in ECO
GW-004: Forward trace a port's driver before insertion
GW-005: Net doesn't exist in design
GW-006: Can't load cell to schematic
GW-007: Trace output pin before delete the gate
GW-008: Can't delete a gate which drives an output port
GW-009: Can't delete a wire which drives an output port
GW-010: Need select a gate to do a operation
GW-011: Can't change ECO gate size
GW-012: No larger size gate in library
GW-013: No smaller size gate in library
GW-014: Connect other side of ECO port first
GW-015: Path is not allowed in port connection
GW-016: Can't disable ECO mode
GW-017: No more ECO operations in undo
GW-018: Need select a pin to do listing endpoints

Warning: Missing argument 1 for pdf_tail(), called in C:\xampp\htdocs\content.php on line 90 and defined in **C:\xampp\htdocs\plib\nand.php** on line **149**